MASTER'S DEGREE THESIS

Master of Science in Advanced Mathematics and Mathematical Engineering

Title: Applications of the Sinkhorn-Knopp algorithm for phylogenetic inference.

Author: Clara Mateo Campo

Advisor: Jesús Fernández Sánchez

Department: Departament de Matemàtica Aplicada I

Academic year: 2015-2016

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya

Facultat de Matemàtiques i Estadística

Master's Degree Thesis

Applications of the Sinkhorn-Knopp algorithm for phylogenetic inference

Clara Mateo Campo

Advisor: Jesús Fernández Sánchez

Departament de Matemàtica Aplicada I - UPC

Abstract

MSC 2010: 92D15, 92D20, 15-04, 15B99, 60J20, 62P10

Keywords: phylogenetic tree, quartet tree, topology reconstruction, general Markov model, Sinkhorn-Knopp algorithm, flattening matrix.

Recent theoretical results relate the rank of certain matrices with the topology of phylogenetic trees over a set of species. In this project, we expect to study how normalization of these matrices (Sinkhorn-Knopp algorithm), that do not affect their rank, can be useful to design phylogenetic reconstruction methods. We also design tests and simulations in order to compare the performance of the new methods with the classical phylogenetic inference methods.

Contents

Chapter 1

Introduction

DNA sequences store all the biological information refering to living organisms and virus. Hence, we could compare DNA sequences with the purpose of relating different species and obtaining more information of their evolutionary history. The evolutionary relationships among species are usually represented by using a phylogenetic tree: each leaf corresponds to a species and the edges represent the evolutionary path. Phylogenetic inference tries to recover, given a set of species, the phylogenetic tree topology that fits the data evolutionary history better.

Phylogenetic inference can be approach from different branches of Mathematics. In the last years, a new and original approach for phylogenetic inference based on numerical linear algebra has been developed. In [7], N. Eriksson suggests to infer the topology of the tree related to a number of species by estimating the distance of certain matrices (called *flattenings*, see Definition 2.4.6) to the space of matrices of rank ≤ 4. This method, which will be referred in this project as `ErikSVD`, relies on the fact that the rank of these matrices is enough (at least, theoretically) to infer the topology.

In a more recent paper [4], J. Fernández-Sánchez and M.Casanellas take advantage of the idea of Eriksson and introduce a variation of `ErikSVD` specific for quartets. The main idea is to reconstruct the topology of the tree by first normalizing the flattening matrices by rows and by columns, and then taking the average value of the distances of these matrices to the space of matrices of rank ≤ 4. This normalization is convenient in order to cope with certain situations where wrong topologies may score really high in the previous method. This new method is called `Erik+2`.

Following the ideas of these two works and based on an old procedure known as the Sinkhorn-Knopp algorithm (see [17]), the main goal of this project is to introduce a new method for phylogenetic reconstruction and study its performance when applied to reconstruct trees of four species. The Sinkhorn-Knopp algorithm is a procedure that allows, under some assumptions, to normalize a matrix by both rows and columns at the same time without modifying the rank. Namely, given a matrix A it produces a new matrix $\tilde{A}$ with the same rank and satisfying that both rows and columns sum up to one. For this project, we focus on the case of 4-leaf trees (quartets) because these are the smallest buildings blocks of phylogenetic reconstruction. That is, there exists a number of methods, known as *quarted-based methods* (see for example [15]) that allow to reconstruct big phylogenetic trees by binding 4-leaves trees together. For this reason, quartet trees make an excellent choice to test and develop new method for phylogenetic reconstruction.

This memoir is divided in five chapters. After this introduction, Chapter 2 is devoted to re-

view the basic concepts of phylogenetics as well as some methods for phylogenetic reconstruction. In Chapter 3, we focus on the theoretical results about the convergence of the Sinkhorn-Knopp algorithm in general, out of the phylogenetic framework. More precisely, Theorems 3.1.5 and 3.1.10 provide conditions that guarantee the convergence of the procedure. We illustrate these results by means of examples and provide a simplification of the proof of Theorem 3.1.5. In Chapter 4, we test the performance of a first method based on the Sinkhorn-Knopp algorithm. To this aim, we consider a tree space previously defined in [11], which allows us to study the accuracy of the method in terms of the branch lengths of the tree. From this study, we derive consequences that allow us, in Chapter 5, to design more sophisticated methods. To this aim, we have taken into account different aspects as the convergence of the Sinkhorn-Knopp algorithm or the relative position of the scores of the matrices obtained throughout this algorithm. The new methods are also tested in the same treespace and for general trees, where the branch lengths are taken randomly and are not subjected to the restrictions of the above tree space. Finally, we compare the performance of these new methods with the most popular methods for phylogenetic reconstruction: neighbour-joining and maximum-likelihood.

Besides the work described in this memoir, a good amount of work carried out was the programming in `C++` of the different methods described above.

Chapter 2

Preliminaries

In this chapter, we introduce the basic concepts of phylogenetics, such as phylogenetic tree, the topology of a tree, evolutionary models and different methods for phylogenetic reconstruction.

2.1 Biological preliminaries

In biology ([9]), phylogenetics is the study of the evolutionary history, development and relationships among a collection of organisms. The similarity of molecular mechanisms among different species suggests that all organisms on Earth had a common ancestor, thus there exists relationships between any organisms. Phylogenies are inferred from all kind of data (morphological, ecological, etc.). In this project, we focus on genetic data, i.e., DNA sequences.

DNA is a molecule that encodes the biological information of an individual. DNA is composed of simpler units called *nucleotides*. There are four different nucleotides: cytosine (`C`), guanine (`G`), adenine (`A`) and thymine (`T`). DNA is a double helix of nucleotides strands (see Figure 2.1) built up according to base pairing rules: each type of nucleotide on one strand bonds with just one type of nucleotide on the other strand (`A` with `T` and `C` with `G`). So, actually, taking only one strand of the DNA it is possible to recover all the information. That is, we work with sequences of thousands of these four nucleotides as `AAGTCGACT`....

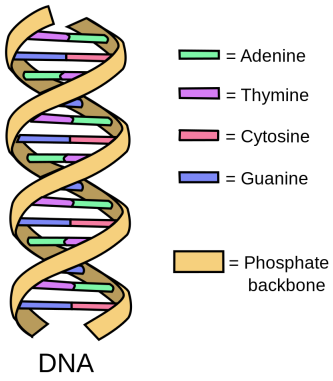


Figure 2.1: DNA structure

In order to study the relationship between two different species we need to compare their DNA sequences. This is a really complex problem: there are extraordinary differences in genome size with billions of base pares of DNA. Furthermore, a big part of these sequence is non-coding. So, the first part of biological sequence analysis consists on finding regions of the two (or more) DNA sequences that are related: namely, we look for some evidence that they have diverged from a common ancestor by a process of mutation. The basic mutational processes are substitutions (one nucleotide is substituted by another one), insertions (a collection of nucleotides are inserted) and deletions (a number of nucleotides are deleted). In this project we will focus on the case of substitutions and no insertions or deletions are considered. An *alignment* is an arrangement of sequences of DNA that identify regions of the sequences that are related (see Table 2.1). This is done with scoring systems, algorithms to find the optimal alignment and statistical methods to evaluate the significance of an alignment score (see [6]). In this project, we work with alignments and we do not focus on the process of generating them.

Gorilla Gorilla	`AACTTCGAGGCTTACCGCTG`
Homo Sapiens	`AACGTCTATGCTCACCGATG`
Pan Troglodytes	`AAGGTCGATGCTCACCGATG`

Table 2.1: Multiple sequence alignment of DNA sequences.

2.2 Phylogenetic trees

In order to represent phylogenys graphically, we usually use a *tree* (see Figure 2.2). In this section, we define all the concepts related to phylogenetic trees.

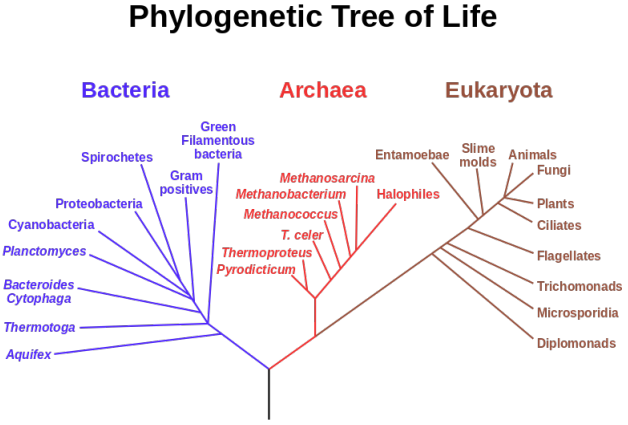


Figure 2.2: Tree of life.

Definition 2.2.1 (Graph, tree, rooted tree, phylogenetic tree)**.** A *graph* G is a pair (V, E) where V is a set of *vertices* and E is a set of *edges* (also called *branches*) between two vertices. The *degree* of a vertex $v \in V$ is the number of edges attached to it.

A *tree* is a connected graph without simple cycles. In a tree, the *leaves* are the vertices with degree 1, while the other vertices are called *internal vertices*. Given a tree $\mathcal{T} = (V, E)$, the set

of leaves of $\mathcal{T}$ is denoted by $\mathcal{L}(\mathcal{T})$. We say that a tree $\mathcal{T}$ is *rooted* if there is one vertex that has been designated as *root* and the edges have an orientation away from the root.

A *trivalent tree* is a tree for which each internal vertex has degree 3.

A *phylogenetic tree* $\mathcal{T}$ *over a set of taxa* (or species) $\mathcal{X}$ is a pair $(\mathcal{T}, \phi)$ where each leaf corresponds to a species, i.e., there is a bijection $\phi : \mathcal{X} \to \mathcal{L}(\mathcal{T})$. If $|\mathcal{X}| = n$ we usually denote the set $\mathcal{X}$ as $\{1, 2, \ldots, n\}$ and the set of trivalent trees with n leaves by $\mathcal{T}_n$.

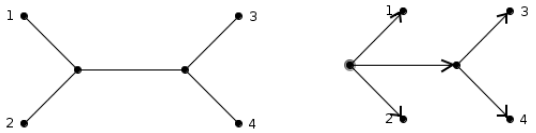


Figure 2.3: Unrooted tree (left), rooted tree (right).

The tree represents the evolutionary history of the species, i.e., two leaves are connected to a same internal vertex if they have evolved from the species represented by that vertex. Then, if the tree is rooted, the root represents the common ancestor of $\mathcal{X}$ and the oriented path from the root to a leaf represents the evolutionary path that the species has gone over.

Definition 2.2.2 (Tree topology)**.** The *tree topology* of a phylogenetic tree is the topology of the tree as a labeled graph. We say that two trees $\mathcal{T}$ and $\mathcal{T}'$ with the same set of leaves $\mathcal{X}$ are *topologically equivalent* if they have the same topology, i.e., there exists an one-to-one correspondence φ between their vertices that respects adjacency and their leaf labeling.

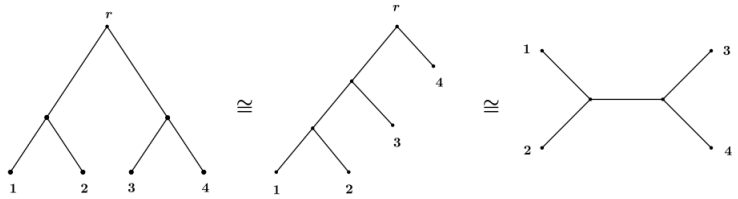


Figure 2.4: Different phylogenetic trees with the same topology as unrooted trees.

Remark 2.2.3. In this project, we will basically work with unrooted trees with 4 leaves, i.e., $\mathcal{X} = \{1, 2, 3, 4\}$. There are three possible topologies of $\mathcal{T}_4$ (see Figure 2.5). We denote them by $\mathcal{T}_{12|34}$, $\mathcal{T}_{13|24}$ and $\mathcal{T}_{14|24}$.

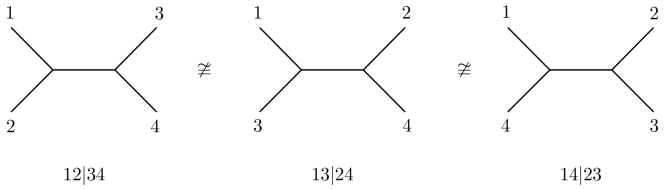


Figure 2.5: Topologies of $\mathcal{T}_4$.

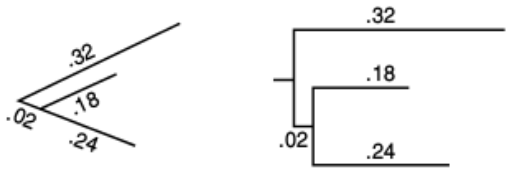


Figure 2.6: Different representations of the same metric tree (from [1]).

Besides the topological structure of a tree, phylogenetics trees may have a metric structure (see Figure 2.6).

Definition 2.2.4 (Metric trees)**.** A *metric tree* is a tree with a metric structure: each edge has assigned a certain length.

In general, the length of an edge represents, somehow, how many DNA mutations occurred along the evolutionary process represented by the edge. That is, the longer an edge is, the more DNA sequence mutations occurred on the time of evolution that the edge represents. Hence, in order to compare differences between DNA sequences it is useful to talk about their distance. A distance is a map which assigns a positive real number to each pair of DNA sequences.

Given two DNA sequences, we can define a distance between them in many ways (see [1]).

Definition 2.2.5 (Jukes-Cantor Distance)**.** The *Jukes-Cantor distance* between DNA sequences S_0 and S_1 is

$$d_{JC}(S_0, S_1) = -\frac{3}{4}\ln\left(1 - \frac{4}{3}p\right)$$

where p is the fraction of sites that disagree in comparing S_0 and S_1.

Solving the above equality for p, we obtain that

$$p = \frac{3}{4}\left(1 - e^{-\frac{4}{3}d(S_0,S_1)}\right).$$

It follows that when the distance grows to infinity, the fraction of sites of disagreement between S_0 and S_1 tends to $\frac{3}{4}$. This observation is the reason of a phenomenon called *long-branch attraction* in phylogenetics. We will come back to it in Chapter 4.

Definition 2.2.6 (Paralinear distance ([10]))**.** Given two DNA sequences S_0 and S_1, the *paralinear distance* is defined by

$$d_P(S_0, S_1) = -\frac{1}{4}\log\det(M)$$

where M is a 4×4 matrix that summarizes the relative frequences of bases in a given pairwise composition. If we denote by n_{ij} the relative frequence where the nucleotide i in S_0 changes to nucleotide j in S_1, we can write M as

$$M = \begin{array}{c} \\ \mathtt{A} \\ \mathtt{C} \\ \mathtt{G} \\ \mathtt{T} \end{array} \begin{array}{c} \begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \end{array} \\ \begin{pmatrix} n_{\mathtt{AA}} & n_{\mathtt{AC}} & n_{\mathtt{AG}} & n_{\mathtt{AT}} \\ n_{\mathtt{CA}} & n_{\mathtt{CC}} & n_{\mathtt{CG}} & n_{\mathtt{CT}} \\ n_{\mathtt{GA}} & n_{\mathtt{GC}} & n_{\mathtt{GG}} & n_{\mathtt{GT}} \\ n_{\mathtt{TA}} & n_{\mathtt{TC}} & n_{\mathtt{TG}} & n_{\mathtt{TT}} \end{pmatrix} \end{array}.$$

Remark 2.2.7. When all non-diagonal entries n_{ij} have the same value (equal to $p/3$), it is easy to check that the paralinear distance reduces to the Jukes-Cantor distance.

2.3 Evolutionary models

In order to model evolutionary processes among a collection of species, the following hypothesis are usually assumed:

1. Mutations in DNA occur randomly.
2. Evolutionary processes at different edges are independent.
3. Sites in the DNA sequence are independent and identically distributed (iid). This means that nucleotides in a DNA sequence do not depend on the other nucleotides in the sequence.

With these hypothesis and the description of a phylogenetic tree, we associate a discrete random variable X_i to each vertex i of $\mathcal{T}$. Each X_i can take κ possible states. We denote the set of the possible values of X_i by $\mathcal{K}$. For instance, if we consider a rooted tree $\mathcal{T}$, with root r, there is an associated vector X_r that is usually denoted by $\pi = (\pi_1, \ldots, \pi_\kappa)$ where $\pi_i \geq 0$ for all $i \in \{1, \ldots, \kappa\}$ and $\sum_{i=1}^{\kappa} \pi_i = 1$. Hence, π is the probability distribution in the alphabet $\mathcal{K}$.

In this project, we will consider $\kappa = 4$ because DNA molecules are essentially sequences of 4 nucleotides and hence $\mathcal{K} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. X_i will represent the value observed at each site of the DNA sequence of the species of the i-th vertex. Since the ancestral species are unknown, we have no information about the internal vertices of the tree: the interior vertices are hidden. Furthermore, the root of the tree represents the common ancestor to all the species represented at the leaves, $\pi = (\pi_{\mathtt{A}}, \pi_{\mathtt{C}}, \pi_{\mathtt{G}}, \pi_{\mathtt{T}})$.

Definition 2.3.1 (Transition Matrix)**.** A *transition matrix* is a $\kappa \times \kappa$ matrix M_e, associated to each edge of a phylogenetic tree. Every entry is the conditional probability $P(x|y, e)$ that the state y at the parent vertex of e is being substituted by the state x at its child, during the evolutionary process along branch e. Since each row are the probabilities of the κ possible changes that can occur in an evolutionary process, the rows of M_e sum to 1.

Remark 2.3.2. If $\kappa = 4$ and $\mathcal{K} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ then the (i, j)-entry of M_e stands for the conditional probability that if nucleotide i occurs at one site of the DNA sequence in the parent vertex on the edge e, then nucleotide j occurs at the descendent vertex at the same site. In this case, the transition matrices have the form

$$M_e = \begin{array}{c} \\ \mathtt{A} \\ \mathtt{C} \\ \mathtt{G} \\ \mathtt{T} \end{array} \begin{array}{c} \begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \end{array} \\ \begin{pmatrix} P(\mathtt{A}|\mathtt{A}, e) & P(\mathtt{C}|\mathtt{A}, e) & P(\mathtt{G}|\mathtt{A}, e) & P(\mathtt{T}|\mathtt{A}, e) \\ P(\mathtt{A}|\mathtt{C}, e) & P(\mathtt{C}|\mathtt{C}, e) & P(\mathtt{G}|\mathtt{C}, e) & P(\mathtt{T}|\mathtt{C}, e) \\ P(\mathtt{A}|\mathtt{G}, e) & P(\mathtt{C}|\mathtt{G}, e) & P(\mathtt{G}|\mathtt{G}, e) & P(\mathtt{T}|\mathtt{G}, e) \\ P(\mathtt{A}|\mathtt{T}, e) & P(\mathtt{C}|\mathtt{T}, e) & P(\mathtt{G}|\mathtt{T}, e) & P(\mathtt{T}|\mathtt{T}, e) \end{pmatrix} \end{array}.$$

According to the shape of the transition matrices one has different *models*. In this project we will basically work with two of them: general Markov model and the generalised Time-Reversible model.

Definition 2.3.3 (General Markov Model)**.** The *General Markov model (GMM)* is the model with no restriction neither in π nor the transition matrices M_e. Then $\pi = (\pi_{\mathtt{A}}, \pi_{\mathtt{C}}, \pi_{\mathtt{G}}, \pi_{\mathtt{T}})$ such that $\sum_i \pi_i = 1$ and

$$M_e = \begin{pmatrix} a_e & b_e & c_e & d_e \\ e_e & f_e & g_e & h_e \\ j_e & k_e & l_e & m_e \\ n_e & o_e & p_e & q_e \end{pmatrix}, \text{ where } \begin{cases} a_e + b_e + c_e + d_e = 1 \\ e_e + f_e + g_e + h_e = 1 \\ j_e + k_e + l_e + m_e = 1 \\ n_e + o_e + p_e + q_e = 1 \end{cases}$$

Definition 2.3.4 (Generalised Time-Reversible [21])**.** The generalised Time-Reversible *(GTR)* is the model with no restriction on π ($\pi = (\pi_{\mathtt{A}}, \pi_{\mathtt{C}}, \pi_{\mathtt{G}}, \pi_{\mathtt{T}})$ such that $\sum_i \pi_i = 1$) and

$$M_e = e^{Qt_e},$$

where Q is a rate matrix with the form

$$Q = \begin{pmatrix} -(a_e + b_e + c_e)\pi_{\mathtt{A}} & a_e\pi_{\mathtt{C}} & b_e\pi_{\mathtt{G}} & c_e\pi_{\mathtt{T}} \\ a_e\pi_{\mathtt{A}} & -(a_e + d_e + f_e)\pi_{\mathtt{C}} & d_e\pi_{\mathtt{G}} & f_e\pi_{\mathtt{T}} \\ b_e\pi_{\mathtt{A}} & d_e\pi_{\mathtt{C}} & -(b_e + d_e + g_e)\pi_{\mathtt{G}} & g_e\pi_{\mathtt{T}} \\ c_e\pi_{\mathtt{A}} & f_e\pi_{\mathtt{C}} & g_e\pi_{\mathtt{G}} & -(c_e + f_e + g_e)\pi_{\mathtt{T}} \end{pmatrix}$$

and where $t_e \geq 0$ represents the time.

Definition 2.3.5 (Evolutionary model on a phylogenetic tree)**.** Given a tree $\mathcal{T}$, an *evolutionary model on* $\mathcal{T}$ is a correspondence between $E(\mathcal{T})$ and the set of transition matrices of the evolutionary model considered and some distribution $\pi = (\pi_{\mathtt{A}}, \pi_{\mathtt{C}}, \pi_{\mathtt{G}}, \pi_{\mathtt{T}})$ in the root. That is, for each $e \in E(\mathcal{T})$ one transition matrix M_e is assigned.

Example 2.3.6. The Figure 2.7 represents an evolutionary model on a phylogenetic tree with the random variables that have been defined in this section. The number of parameters of a statistical model in a phylogenetic tree depends on the chosen model and on the tree. For instance, if the model of this tree is the General Markov model, we have 3×4 free parameters for each substitution matrix and 3 free parameters for the vector π_r. Therefore, this tree evolving under the GMM has $3 \cdot 4 \cdot 6 + 3 = 75$ parameters.

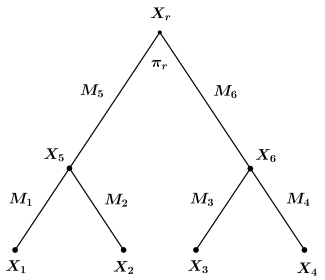


Figure 2.7: Statistical model on a rooted phylogenetic 4-leaved tree.

2.4 Joint distribution and flattenings

We fix now an evolutionary model $\mathcal{M}$ on a n-leaf tree $\mathcal{T}$ rooted at a vertex r. In what follows we describe how to compute the joint probability of observing states $x_1, x_2, \ldots, x_n$ at the leaves according to the Markov process we have described.

We denote by $p_{x_1,\ldots,x_n}$ the joint distribution at the leaves of $\mathcal{T}$, which means that $p_{x_1,\ldots,x_n}$ is the probability that the random variables $X_1, \ldots, X_n$ of the leaves take the states $x_1, \ldots, x_n$:

$$p_{x_1,\ldots,x_n} = Prob(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n).$$

We denote by P as a κ^n-dimensional vector, whose entries are the joint probabilities $p_{x_1\ldots x_n}$,

$$P = (p_{x_1\ldots x_n})_{x_1\ldots x_n \in \mathcal{K}}.$$

Under the Markov assumption and since the evolutionary processes are independent of each other and just depend on a common node we can express $p_{x_1,\ldots,x_n}$ in terms of the entries of the substitution matrices

$$p_{x_1,\ldots,x_n} = \sum_{x_r,(x_v)_{v\in Int(\mathcal{T})}} \prod_{e\in E(\mathcal{T})} M_e(x_{a(e)}, x_{d(e)}), \tag{2.1}$$

where $x_r \in \mathcal{K}$ is a state of the root, $x_{a(e)} \in \mathcal{K}$ is a state of the parent vertex of the edge e, and $x_{d(e)} \in \mathcal{K}$ is the state of the child node of the edge e. If e is a terminal edge then $x_{d(e)} = x_i$, for some i. Every entry of P can be seen as a polynomial on the parameters of the model $\mathcal{M}$ as variables.

Example 2.4.1. We compute now the joint distribution p_{x_1,x_2,x_3,x_4} of the tree represented in Example 2.3.6. Using equation 2.1 we get

$$p_{x_1,x_2,x_3,x_4} = \sum_{x_r\in\mathcal{K}} \sum_{x_5\in\mathcal{K}} \sum_{x_6\in\mathcal{K}} \pi_{x_r} \cdot M_5(x_r, x_5) \cdot M_1(x_5, x_1) \cdot M_2(x_5, x_2) \cdot \\ \cdot M_6(x_r, x_6) \cdot M_3(x_6, x_3) \cdot M_4(x_6, x_4)$$

Definition 2.4.2 (Bipartition)**.** Given a finite set $\mathcal{X}$, a *bipartition* $A \mid B$ of $\mathcal{X}$ are two sets A and B with $|A| \geq 2$ and $|B| \geq 2$ such that $A \cup B = \mathcal{X}$ and $A \cap B = \emptyset$.

Example 2.4.3. If $\mathcal{X} = \{1, 2, 3, 4\}$ we can consider $A = \{1, 2\}$ and $B = \{3, 4\}$.

If $\mathcal{T}$ is a phylogenetic tree, removing and interior edge e from $\mathcal{T}$ induces a bipartition of $\mathcal{L}(\mathcal{T})$. Such a bipartition is referred as the *split associated to the edge* e. Notice that whether a given bipartition $A \mid B$ is a split or not depends on the topology of the tree.
Buneman stated in [3] the following result

Theorem 2.4.4 (Buneman's Theorem)**.** *Let $\mathcal{T}$ be a trivalent tree with n leaves. Then, the whole topology of the tree $\mathcal{T}$ can be recovered from the collection of its $2n-3$ splits.*

Example 2.4.5. Consider the tree represented in Figure 2.8. The red edges are internal and, furthermore, removing any of them induces a bipartition of $\{1, 2, \ldots, 7\}$. For instance, if we remove the leftmost internal edge we obtain the bipartition $12 \mid 34567$.

This example allows us to illustrate the Buneman's Theorem: we can recover the topology of the tree represented in Figure 2.8 by considering the set of the splits $\{12 \mid 34567, 123 \mid 4567, 67 \mid 12345, 45 \mid 12367\}$. The split $12 \mid 34567$ informs us that the leaves 1 and 2 are children of the same internal vertex and that there is an internal edge from this vertex to the rest of the tree. The next split, $123 \mid 4567$ tells us that there is another internal vertex (parent of 3) and from this vertex to the rest of the tree 4567 an internal edge. Following this procedure we recover the tree.

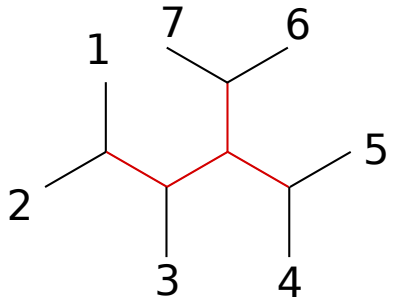


Figure 2.8: A trivalent tree topology. Internal edges are represented by red and induce the splits associated to this topology.

Another important concept in this work is the flattening along a bipartition:

Definition 2.4.6 (Flattening along a bipartition)**.** Given a tree $\mathcal{T}$ and a bipartition $A|B$ of $\mathcal{L}(\mathcal{T})$, we denote by $\tilde{X}_A$ and $\tilde{X}_B$ the random variables associated to A and B. Then $\tilde{X}_A$ and $\tilde{X}_B$ can take $\kappa^{|A|}$ and $\kappa^{|B|}$ states respectively. We define the *flattening* along $A \mid B$ associated to P, $Flat_{A|B}(P)$, as a $\kappa^{|A|} \times \kappa^{|B|}$ matrix whose entries are the joint distribution of the observations of $\tilde{X}_A$ and $\tilde{X}_B$:

$$Flat_{A|B}(P) = \begin{array}{c} \\ \text{States of } \tilde{X}_A \end{array} \begin{array}{c} \text{States of } \tilde{X}_B \\ \begin{pmatrix} p_{u_1v_1} & p_{u_1v_2} & \cdots & p_{u_1v_{\kappa^{|B|}}} \\ p_{u_2v_1} & p_{u_2v_2} & \cdots & p_{u_2v_{\kappa^{|B|}}} \\ \vdots & \vdots & \ddots & \vdots \\ p_{u_{\kappa^{|A|}}v_1} & p_{u_{\kappa^{|A|}}v_2} & \cdots & p_{u_{\kappa^{|A|}}v_{\kappa^{|B|}}} \end{pmatrix} \end{array}.$$

Example 2.4.7 (Flattening along a bipartition on 4 taxa)**.** Let $\mathcal{T}$ be a tree with 4 leaves. Let $\kappa = 4$ and $\mathcal{K} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. The bipartition $13 \mid 24$ induces the 16×16 matrix where the rows are indexed by the states of taxa 1 and 3 and the columns by the states of taxa 2 and 4:

$$Flat_{13|24}(P) = \begin{array}{c} \\ \text{States at leaves 1 and 3} \end{array} \begin{array}{c} \text{States at leaves 2 and 4} \\ \begin{pmatrix} p_{\mathtt{AAAA}} & p_{\mathtt{AAAC}} & p_{\mathtt{AAAG}} & \cdots & p_{\mathtt{AATT}} \\ p_{\mathtt{ACAA}} & p_{\mathtt{ACAC}} & p_{\mathtt{ACAG}} & \cdots & p_{\mathtt{ACTT}} \\ p_{\mathtt{AGAA}} & p_{\mathtt{AGAC}} & p_{\mathtt{AGAG}} & \cdots & p_{\mathtt{AGTT}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{\mathtt{TTAA}} & p_{\mathtt{TTAC}} & p_{\mathtt{TTAG}} & \cdots & p_{\mathtt{TTTT}} \end{pmatrix} \end{array}.$$

There are 3 flattenings on 4 taxa that we will consider afterwards: $Flat_{12|34}(P)$, $Flat_{13|24}(P)$ and $Flat_{14|23}(P)$ (see figure 2.5).

Coming up next, we introduce an important result by N. Eriksson concerning the rank of the flattening matrices. This theorem will be relevant in this project.

Theorem 2.4.8 ([7], Theorem 2.35, for a complete proof see [4].)**.** *Let $\mathcal{T}$ be a trivalent tree with $|\mathcal{L}(\mathcal{T})| = n$, and assume that the joint probability distribution P arises from $\mathcal{T}$ under the general Markov model with an alphabet with κ letters. Let $A \mid B$ be a bipartition of $\mathcal{L}(\mathcal{T})$. Then:*

1. *If $A \mid B$ is a split of $\mathcal{T}$, the generic rank of $Flat_{A|B}(P)$ is κ.*

2. *If $A \mid B$ is not a split of $\mathcal{T}$, the generic rank of $Flat_{A|B}(P)$ is at least κ^2.*

Example 2.4.9. Consider $\mathcal{K} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ and let $\mathcal{T}$ be a tree with leaves. Let P be a generic distribution arising from $\mathcal{T}$ under the GMM. According to Theorem 2.4.8, the 16×16 matrix $Flat_{A|B}(P)$ has rank 4 if the bipartition $A \mid B$ occurs in the tree. Otherwise, it has rank 16.

2.5 Methods for phylogenetic reconstruction

In this section some popular methods for phylogenetic reconstruction are described. The main goal of these methods is, given DNA alignments of the taxa that we want to relate, to reconstruct the phylogenetic tree that represents the evolutionary process of these taxa. Notice that there is no information relative to the internal vertices since we do not yet know the tree topology.

2.5.1 Neighbor-Joining

Distance methods try to reconstruct a phylogenetic tree using information that we believe describes the total distances between the DNA sequence (in the sense explained in Section 2.2). Roughly speaking, distance methods try to make the closest sequences into neighbors in the tree. Among distance methods, the most popular is Neighbor-Joining ([16] and [20]).

Assume that we have n DNA sequences: $S_1, S_2, \ldots, S_n$. After fixing a way to compute the distance between DNA sequences, we compute the distances between all pairs of data: $d(S_1, S_2) = d(S_2, S_1)$, $d(S_1, S_3), \ldots, d(S_{n-1}, S_n)$.
Then, two sequences (let us say, S_l and S_k) are taken as neighbors in the phylogenetic tree if the 4-*point condition* holds, i.e.,

$$d(S_l, S_k) + d(S_i, S_j) < d(S_l, S_i) + d(S_k, S_j), \quad \forall i, j \notin \{l, k\}.$$

Neighbor-Joining is a clustering method designed so that this condition holds at every step of the algorithm (for a complete description of the algorithm see [16]).
Compared with other methods, distance methods are really fast when we deal with a big amount of data. However, since they only use a distance measure to construct the tree, one could argue that they do not use all the information available.

2.5.2 Maximum Likelihood

Another approach to phylogenetic reconstruction is the Maximum Likelihood method. This method assumes a particular evolutionary model (see section 2.3 for some examples) and considers a specific tree relating the data as a candidate. Then, assuming that the tree is correct, it computes the *likelihood* of that tree. That is, the probability that the DNA sequences of our data could have been produced from the fixed tree with this evolutionary model. If this likelihood is computed for all possible trees, then the method chooses the tree with the greatest likelihood value as the tree that fits our data better.
The main disadvantage of this method is that is computational expensive. That is because the number of trees fitting n sequences of data grows factorially. Hence, it becomes unfeasible to reconstruct trees with a lot of leaves.

2.5.3 `ErikSVD` for quartets

Eriksson in [7] proposed a new topology reconstruction method based on the work on invariants of Allman and Rhodes [2]. Below, we explain the key concepts and results on which is based the method. Following [8], we will refer this method as `ErikSVD`.
Based on Theorem 2.4.8, `ErikSVD` tries to find out which bipartition correspond to splits of the correct tree, that is, have flattenings with rank nearest to 4 (for DNA sequences, $\kappa = 4$). Hence, we need to choose a metric that allows to compute the distance between a matrix and the space of rank 4 matrices. For this purpose, we use the Singular Value Decomposition of a matrix and the Frobenius norm.

Definition 2.5.1 (Frobenius norm)**.** Let $A = (a_{ij})$ be an $m \times n$ matrix. The *Frobenius norm*, written $||A||_F$, is the root-sum-of-squares norm on $\mathbb{R}^{m \cdot n}$. That is

$$||A||_F = \sqrt{\sum a_{ij}^2}.$$

Definition 2.5.2 (Singular Value Decomposition)**.** A *singular value decomposition* of an $m \times n$ matrix A (with $m \geq n$) is a factorization $A = U\Sigma V^T$ where U is $m \times n$ and satisfies $U^T U = I$, V is $n \times n$ and satisfies $V^T V = I$ and $\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_n)$, where $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_n \geq 0$ are called the *singular values* of A.

The following result is crucial for us.

Theorem 2.5.3 ([5])**.** *The distance from A to the nearest rank k matrix is*

$$d_k(A) := \min_{Rank(B)=k} ||A - B||_F = \sqrt{\sum_{i=k+1}^{m} \sigma_i^2}.$$

`ErikSVD` is a phylogenetic reconstruction method that uses this result as a criterion to decide whether a given bipartition is a split of the tree. In this context, if A is the flattening matrix along a bipartition (see Definition 2.4.6), the value of $d_4(A)$ will be called *the Frobenius score* of that bipartition. Combined with Buneman's theorem (see Theorem 2.4.4), it allows to determine the right topology of the tree.

Remark 2.5.4. Eriksson proposed a general version of the algorithm for a multiple alignment of genomic data from n species and m possible states. However, reduced to our case of interest, and according to the situation described in Example 2.4.9, the method reduces to deal with the three possible topologies of Figure 2.5.

Algorithm 2.5.5 (`ErikSVD` for quartets)**.** Given a multiple alignment of 4-taxon data, let P the point of joint distributions obtained from the alignment. Compute the Frobenius score of each bipartition, i.e. $d_4(Flat_{12|34}(P))$, $d_4(Flat_{13|24}(P))$ and $d_4(Flat_{14|23}(P))$. Choose the tree topology corresponding to the flattening with the minimum score.

2.5.4 `Erik+2`

In [8], a new topology reconstruction method is introduced: `Erik+2`. This method is based on `ErikSVD`. `Erik+2` tries to overcome a couple of drawbacks that are explained below. On one hand, it is well known that when the tree has long branches the reconstruction methods have a worst performance. This phenomenon is known as *long branch attraction* (see Section 4.1 for a more details). On the other hand, the presence of short branches in the tree is reflected in the distribution of the nonzero entries of certain flattening matrices: the highest entries in the matrix are distributed in the columns that represent that no mutation has been occurred. This fact can be corrected by normalizing the matrix by both rows and columns as described in Step 2 of the next algorithm (see [8] for further details).

Remark 2.5.6. An easy but crucial point is that normalizing a matrix (by rows or columns) does no affect the rank of the matrix, but affects its singular values and so, the Frobenius score defined in the previous section (see Figure 2.9).

Algorithm 2.5.7 (`Erik+2`)**.** The algorithm has the following steps:

- **Step 1**: Compute $Flat_{12|34}(P)$, $Flat_{13|24}(P)$ and $Flat_{14|23}(P)$.
- **Step 2**: For each matrix $Flat_{A|B}(P)$, normalize it by rows obtaining $(Flat_{A|B}(P))_r$. Also normalize it by columns obtaining $(Flat_{A|B}(P))_c$.
- **Step 3**: Assign to each flattening the score

$$sc(Flat_{A|B}(P)) = \frac{d_4((Flat_{A|B}(P))_r) + d_4((Flat_{A|B}(P))_c)}{2}.$$

- **Step 4**: Choose the bipartition with the minimum score.

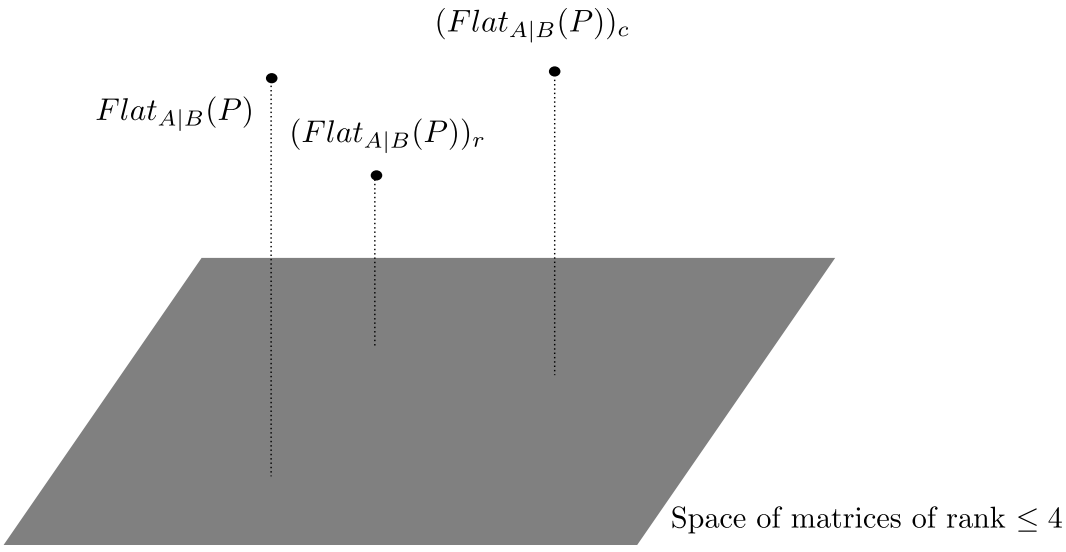


Figure 2.9: Representation of the Frobenius score for the matrices $Flat_{A|B}(P)$, $(Flat_{A|B}(P))_r$ and $(Flat_{A|B}(P))_c$.

Chapter 3

The Sinkhorn-Knopp algorithm

In the last section of the previous chapter, we described the `Erik+2` method. The idea of this method is to first balance the flattening matrix normalizing by columns and rows. A different approach would be to normalize first by rows, and then by columns: $A \to A_r \to (A_r)_c$. However, the rows of the matrix $(A_r)_c$ obtained would not be balanced anymore. Similarly, one could also normalize A first by columns and then by rows $A \to A_c \to (A_c)_r$, but we would have to deal with an analogous situation. This issue seems to be overcomed if we keep on normalizing by rows and columns:

$$\begin{aligned} &A \to A_r \to (A_r)_c \to ((A_r)_c)_r \to \cdots \qquad \text{(starting by rows)} \\ &A \to A_c \to (A_c)_r \to ((A_c)_r)_c \to \cdots \qquad \text{(starting by columns)} \end{aligned}$$

since, under some assumptions, these sequences are known to converge to the same matrix. Based on this idea, in this chapter we are concerned with the convergence of alternately scaling the rows and columns of a square matrix A. This procedure is known as the Sinkhorn-Knopp algorithm ([17]).

3.1 Theoretical results and convergence of the Sinkhorn-Knopp algorithm

In this section, we will discuss some sufficient conditions to guarantee the convergence of the above sequences and we will study what kind of matrix is obtained in this case of convergence. Sinkhorn and Knopp obtained some convergence results for the method in the 1960's (see [18] and [17]).

Algorithm 3.1.1 (Sinkhorn-Knopp algorithm)**.** Given a $n \times n$ square matrix $A = (a_{ij})$, the Sinkhorn-Knopp algorithm consists on alternately scaling the rows and columns of A. That is

- **Scaling A by rows**. Consider for each row i the sum of its elements, $\lambda_i = \sum_{j=1}^{n} a_{ij}$. Then, the result of scaling A by rows is the matrix (a_{ij}/λ_i). Notice that the rows of the matrix (a_{ij}/λ_i) sum up to 1. Indeed, by definition of λ_i, $\sum_{j=1}^{n} a_{ij}/\lambda_i = 1$ for all $i = 1, \ldots, n$.
- **Scaling A by columns**. Consider for each column j the sum of its elements, $\mu_j = \sum_{i=1}^{n} a_{ij}$. Similarly, A scaled by columns is the matrix (a_{ij}/μ_j). Notice that the columns of the matrix (a_{ij}/μ_j) sum up to 1. Indeed, by definition of μ_j, $\sum_{i=1}^{n} a_{ij}/\mu_j = 1$ for all $i = 1, \ldots, n$.

Alternately scaling the rows and columns of A we obtain a sequence of matrices

$$\{A_0 = A, A_1, A_2, A_3, \dots\}.$$

Depending on if we start scaling by columns or rows we can write

$$\{A_0 = A, A_1 = A_c, A_2 = A_{cr}, \dots\} \quad \text{or} \quad \{A_0 = A, A_1 = A_r, A_2 = A_{rc}, \dots\}$$

respectively.

Remark 3.1.2. Each step k of the Sinkhorn-Knopp algorithm consists on scaling the previous matrix A_k by rows or columns. This corresponds to multiply A_k by a diagonal matrix. For rows we multiply $diag(\lambda_1^{-1}, \dots, \lambda_n^{-1}) \cdot A_k$ and for columns $A_k \cdot diag(\mu_1, \dots, \mu_n)$. In any case, the rank of the matrices A_k does not change when applying the algorithm.

Example 3.1.3 (Sinkhorn-Knopp algorithm)**.** Let us consider the matrix

$$A = \begin{pmatrix} 1 & 0.4 & 1.6 & 0.81 \\ 0.2 & 2.3 & 0.5 & 0.7 \\ 1.2 & 0.9 & 2.5 & 3 \\ 0.5 & 1.5 & 4 & 2.6 \end{pmatrix}$$

We will show Sinkhorn-Knopp algorithm starting by columns. Hence, we first compute the sum of the columns:

$$\mu_1 = 2.9, \quad \mu_2 = 5.1, \quad \mu_3 = 8.6, \quad \mu_4 = 7.11.$$

We normalize the matrix A by columns dividing the column j by μ_j for $j = 1, \dots, 4$. We obtain:

$$A_1 = (a_{ij}^1) = \begin{pmatrix} 1/2.9 & 0.4/5.1 & 1.6/8.6 & 0.81/7.11 \\ 0.2/2.9 & 2.3/5.1 & 0.5/8.6 & 0.7/7.11 \\ 1.2/2.9 & 0.9/5.1 & 2.5/8.6 & 3/7.11 \\ 0.5/2.9 & 1.5/5.1 & 4/8.6 & 2.6/7.11 \end{pmatrix} \simeq \begin{pmatrix} 0.344828 & 0.078431 & 0.186047 & 0.113924 \\ 0.068966 & 0.450980 & 0.058140 & 0.098453 \\ 0.413793 & 0.176471 & 0.290698 & 0.421941 \\ 0.172414 & 0.294118 & 0.465116 & 0.365682 \end{pmatrix}$$

The sum of the elements of every row of the new matrix A_1 are:

$$\lambda_1 = 0.723230, \quad \lambda_2 = 0.676538, \quad \lambda_3 = 1.302902, \quad \lambda_4 = 1.297330.$$

Thus,

$$A_2 = (a_{ij}^2) = (a_{ij}^1/\lambda_i) \simeq \begin{pmatrix} 0.479789 & 0.108446 & 0.257244 & 0.157521 \\ 0.101939 & 0.666600 & 0.085937 & 0.145524 \\ 0.317593 & 0.135444 & 0.223115 & 0.323847 \\ 0.132899 & 0.226710 & 0.358518 & 0.281873 \end{pmatrix}$$

Now, we want to know if the sequence of matrices $\{A_0 = A, A_1, A_2, A_3, \dots\}$ obtained alternately scaling the rows and columns of A converge. In an ideal situation, this sequence must converge to a matrix whose rows and columns sum to 1, i.e., to a doubly stochastic matrix.

Definition 3.1.4 (Doubly stochastic matrix)**.** Given a square matrix $A = (a_{ij})$ we say that A is a *doubly stochastic matrix* if all the entries of A are non-negative and each of its rows and columns sums to 1, i.e.

$$\sum_i a_{ij} = \sum_j a_{ij} = 1.$$

There are some useful theorems about the convergence of the sequence $\{A_0 = A, A_1, A_2, \dots\}$ to a doubly stochastic matrices. Sinkhorn proved in [17] the following result for arbitrary positive matrices:

Theorem 3.1.5 ([17], Theorem 2, page 887)**.** *Let $A = (a_{ij})$ be a strictly positive $n \times n$ matrix (i.e. $a_{ij} > 0\ \forall i, j$). The iterative process of alternately normalizing the rows and columns of A is convergent to a strictly positive doubly stochastic matrix.*

Proof. The iteration of Sinkhorn-Knopp algorithm produces a sequence of positive matrices which alternately have row and columns sums one. We can consider the two subsequences which are composed of the matrices with row sums one and the matrices with column sums one. We want to see that the two subsequences converge to the same doubly stochastic matrix. But, actually, only one convergence proof is required: if we prove the convergence of one subsequence, the terms of the other subsequence can be generated by the transposing the terms of the other. We denote by $\{A_k\}_k = \{(a_{kij})\}_k$ the sequence with column sums one. In order to prove our claim, we will proceed in three steps.

- **Step 1**: In this part of the proof, we are going to see that the maximum and minimum column sums are monotone sequences and hence they have limit.
  Let A_k have row sums $\lambda_{k1}, \dots, \lambda_{kn}$, i.e., $\lambda_{ki} = \sum_{j=1}^{n} a_{kij}$ for all i. Its row normalization can be express by $(\frac{a_{kij}}{\lambda_{ki}})$ and

$$A_{k+1} = \left(\frac{a_{kij}}{\lambda_{ki}\mu_{kj}}\right) \text{ where } \mu_{kj} = \sum_i \frac{a_{kij}}{\lambda_{ki}}.$$

  Since A_k is normalized by columns, $\sum_{i,j=1}^{n} a_{ij} = n$, which implies the following inequalities

$$\min(\lambda_{ki}) \leq n \min(a_{kij}) \leq n \cdot \frac{n}{n^2} \leq 1$$

$$\max(\lambda_{ki}) \geq n \max(a_{kij}) \geq n \cdot \frac{n}{n^2} \geq 1$$

  Furthermore, since $\sum_{i=1}^{n} a_{kij} = 1\ \forall j$, every μ_{kj} is a convex combination of the $\{1/\lambda_{ki}\}_{i=1,\dots n}$. Similarly, when we normalize again, we have that

$$\lambda_{k+1,i} = \sum_{j=1}^{n} a_{k+1,ij} = \sum_{j=1}^{n} \frac{a_{kij}}{\lambda_{ki}\mu_{kj}}$$

  and $\sum_{i=1}^{k} a_{kij}/\lambda_{ki} = 1$, which implies that $\lambda_{k+1,i}$ is a convex combination of $\{1/\mu_{kj}\}_{j=1,\dots,n}$. Then, for all k,

$$\min_i \left(\frac{1}{\lambda_{ki}}\right) \leq \mu_{kj} \leq \max_i \left(\frac{1}{\lambda_{ki}}\right) \iff \min_i(\lambda_{ki}) \leq \frac{1}{\mu_{kj}} \leq \max_i(\lambda_{ki})$$

$$\min_j \left(\frac{1}{\mu_{kj}}\right) \leq \lambda_{k+1,i} \leq \max_j \left(\frac{1}{\mu_{kj}}\right)$$

  which implies that

$$\min_i(\lambda_{ki}) \leq \min_i(\lambda_{k+1,i}) \leq 1 \leq \max_i(\lambda_{k+1,i}) \leq \max_i(\lambda_{ki}). \tag{3.1}$$

Similarly,

$$\min_j(\mu_{kj}) \leq \min_j(\mu_{k+1,j}) \leq 1 \leq \max_j(\mu_{k+1,j}) \leq \max_j(\mu_{kj}). \tag{3.2}$$

Therefore the maximum and minimum row and column sums are monotone sequence and hence have limits.

- **Step 2:** We denote by a_k the minimal element of A_k and we first see that $\{a_k\}$ is bounded by an positive value.
If we write $A_1 = (a_{ij})$ realize that, since the columns of A_k sums one, we can write for all j

$$\sum_{i=1}^{n} \frac{a_{ij}}{\lambda_{1i}\lambda_{2i}\cdots\lambda_{ki}\mu_{1j}\mu_{2j}\cdots\mu_{kj}} = 1 \iff \frac{1}{\sum_{i=1}^{n}\frac{a_{ij}}{\lambda_{1i}\lambda_{2i}\cdots\lambda_{ki}}} = \frac{1}{\mu_{1j}\mu_{2j}\cdots\mu_{kj}}$$

Then, for all i and j:

$$\frac{1}{\mu_{1j}\mu_{2j}\cdots\mu_{kj}} \leq \frac{1}{\frac{a_{ij}}{\lambda_{1i}\lambda_{2i}\cdots\lambda_{ki}}} \leq \frac{1}{\frac{a}{\lambda_{1i}\lambda_{2i}\cdots\lambda_{ki}}}$$

where a denotes the minimal element of $A_1 = (a_{ij})$. If we write $x_{ki} = (\lambda_{1i}\lambda_{2i}\cdots\lambda_{ki})^{-1}$ and $y_{kj} = (\mu_{1j}\mu_{2j}\cdots\mu_{kj})^{-1}$, we have that, in particular, $y_{kj} \leq 1/(a\max(x_{ki}))$.
Since

$$\sum_{j=1}^{n} x_{ki}a_{ij}y_{kj} = \lambda_{k+1,i} \geq \min_i(\lambda_{k+1,i}) \geq \min_i(\lambda_1) := \lambda$$

it follows that

$$x_{ki} \geq \frac{\lambda}{\sum_{j=1}^{n} a_{ij}y_{kj}} \geq \frac{a\lambda\max_i(x_{ki})}{n}.$$

Also,

$$y_{kj} = \frac{1}{\sum_{i=1}^{n} a_{ij}x_{ki}} \geq \frac{1}{n\max_i(x_{ki})}.$$

We see that

$$a_{k+1,ij} = x_{ki}a_{ij}y_{nj} \geq \frac{a\lambda}{n^2} := \mu \tag{3.3}$$

which implies that $a_k \geq \mu > 0$ for all k, as we wanted to see.

- **Step 3:** To finish the proof, we are going to see that $\{a_n\}$ have limit one.
From equation (3.1), it is clear that $\max(\lambda_k) \xrightarrow{k} 1 + c$ where $c \geq 0$. We want to prove that, necessarily $c = 0$. Let us write $\max(\lambda_k) = 1 + c_k$. Consider now for each k, $\delta^k_{j,\lambda\leq 1} = \sum a_{kij}$ where the sum is taken over all i for which $\lambda_{ki} \leq 1$, and $\delta^k_{j,\lambda>1} = \sum_{i=1}^{n} a_{ij} - \delta^k_{j,\lambda\leq 1}$. Then,

$$\mu_{kj} = \sum_{i=1}^{n} \frac{a_{kij}}{\lambda_{ki}} \geq \delta^k_{j,\lambda\leq 1} + \frac{1}{1+c_k}\delta^k_{j,\lambda>1} = \frac{\delta^k_{j,\lambda\leq 1} + \delta^k_{j,\lambda>1} + c_k\delta^k_{j,\lambda\leq 1}}{1+c_k} \geq \frac{1+c_ka_k}{1+c_k} \tag{3.4}$$

Then, if we write $\max(\lambda_{k+1,i_0})$ for some i_0, and using the previous expression

$$1 + c \leq \lambda_{k+1,i_0} = \sum_{j=1}^{n} \frac{a_{ki_0j}}{\lambda_{ki_0}\mu_{kj}} \leq \frac{1+c_ka_k}{1+c_k}.$$

Since $c_k \to c$, if $c > 0$ then $a_k \to 0$, which is a contradiction with the result obtained in the equation (3.3). Thus, $c = 0$ and $\max(\lambda_k) \to 1$. Similarly, it can be proved that $(\min(\lambda_k)) \xrightarrow{k} 1$.

□

Example 3.1.6. Consider the matrix A of Example 3.1.3 and $\epsilon = 10^{-5}$. By Theorem 3.1.5, matrix A converges to a doubly stochastic matrix. After 10 iterations we obtain the matrix

$$A_{10} = \begin{pmatrix} 0.460827 & 0.090041 & 0.277391 & 0.171741 \\ 0.109071 & 0.612702 & 0.102585 & 0.175642 \\ 0.302995 & 0.111004 & 0.237481 & 0.348520 \\ 0.127103 & 0.186260 & 0.382542 & 0.304096 \end{pmatrix}$$

which is normalized by rows, and the sum of the columns μ_j satisfies $|\mu_j - 1| < \epsilon$.

Theorem 3.1.5 assumes the matrix A is strictly positive. The next example shows that if some entry is zero, this theorem does not necessarily hold anymore.

Example 3.1.7. Consider the matrix

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Normalizing first by columns and then it by rows we obtain, respectively, the matrices A_1 and A_2:

$$A_1 = \begin{pmatrix} 0 & 0 & 1/2 \\ 0 & 0 & 1/2 \\ 1 & 1 & 0 \end{pmatrix} \quad A_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1/2 & 1/2 & 0 \end{pmatrix}$$

We observe that $A_3 = A_1$. Then, the iteration oscillates between A_1 and A_2 and, hence, in this case, there is no convergence.

Due to the nature of the matrices we are dealing with (flattening on 4 taxa, see Definition 2.4.6), these may have 0 entries. Actually, unless the alignment is extremely long, we should expect some entries of the flattening to be 0. We would like to have some convergence condition for the iterative algorithm for nonnegative matrices. Sinkhorn and Knopp gave a theorem in this direction in [18].
We need first some definitions.

Definition 3.1.8. Let $A = (a_{ij})$ be an $n \times n$ matrix and σ a permutation of $\{1, \ldots, n\}$. The *diagonal of A corresponding to σ* is the sequence of elements $a_{1,\sigma(1)}, a_{2,\sigma(2)}, \ldots, a_{n,\sigma(n)}$. If σ is the identity, the diagonal is called the *main diagonal*.
If A is a nonnegative square matrix, A is said to have *total support* if $A \neq 0$ and every positive element of A lies on a *positive diagonal*, i.e., a diagonal whose elements are all nonnegative.
A nonnegative matrix that contains a positive diagonal is said to *have support*.

Example 3.1.9. Consider the matrix

$$A = \begin{pmatrix} 5 & 0 & 7 \\ 0 & 1 & 4 \\ 0 & 2 & 0 \end{pmatrix}$$

and the permutation $\sigma = (1 \quad 3 \quad 2)$. In this case, the diagonal of A corresponding to σ is 5, 4, 2. Since this diagonal is positive, the matrix A has support. But, the element $a_{1,3} = 7$ does not lie on a positive diagonal, that is, A has no total support.

Theorem 3.1.10 ([18], Theorem on page 344)**.** *Let A be a nonnegative $n \times n$ matrix. A necessary and sufficient condition that the iterative process of alternately normalizing the rows and columns of A will converge to a doubly stochastic limit is that A has support.*

The proof of this theorem is quite technical but the main idea is the same that we have seen in Theorem 3.1.5. For more details see [18].

3.2 Towards a new method of phylogenetic inference

As we have explained, `Erik+2` chooses a tree topology on 4-taxon data according to the one that minimizes the average of the Frobenius score when we normalize the flattening matrices by columns and rows with the aim of balance the matrices. The Sinkhorn-Knopp algorithm applied to the flattening matrices (based on a similar idea) is: we alternately scale the flattenings matrices by columns and rows until it converges (if it satisfies the necessary conditions of Theorem 3.1.10). We either could start this algorithm normalizing by rows or columns. Moreover, `Erik+2` method can be understood as the average of the Frobenius score in the first iteration of the Sinkhorn-Knopp algorithm when we start by rows or columns.
So, a natural question is what happens if, instead of considering the first iteration, we leave the iterations go and we consider the Frobenius score when the method stabilizes (in case of convergence) or when a fixed number of iterations have been considered.
From this idea it arises the following algorithm that will be referred in the sequel as the *Sinkhorn-Knopp method* (SK-method).

Algorithm 3.2.1 (SK-method)**.** Given a 4-taxon alignment, a maximum number n of iterations and a small $\epsilon > 0$,

- **Step 1**: Compute $Flat_{12|34}(P)$, $Flat_{13|24}(P)$ and $Flat_{14|23}(P)$.
- **Step 2**: For each matrix $Flat_{A|B}(P)$, apply the Sinkhorn-Knopp algorithm (Algorithm 3.1.1) to $Flat_{A|B}(P)$ until either it stabilizes for this value of ϵ or until the maximum number of iterations, n, is reached. This can be done by starting either by columns or rows.
- **Step 3**: When the process has either stabilized or reached n iterations, compute the Frobenius distance of the obtained matrix.
- **Step 4**: Choose the bipartition with the minimum Frobenius score after applying the Sinkhorn-Knopp algorithm.

Remark 3.2.2. After Theorem 3.1.10, the necessary and sufficient condition for the convergence of the procedure seems to be too strong and we cannot expect this condition to fulfill for flattening matrices specially when the length of the alignment is relatively short compared with the size of the matrices. This will be confirmed by the experiments described in the following chapter, which study the performance of the SK-method on simulated data.

Remark 3.2.3. Even though we have proposed SK-method for 4-taxon alignment, the results of this chapter work for any dimension of the flattening matrices. Hence, we can apply a general version of SK-method for n species and κ possible states.

Chapter 4

Simulations

In order to compare the different methods for phylogenetic reconstruction, we study their performance on simulated 4-taxon data (see [8]). In this chapter we first describe the simulated data used to study the performance and accuracy of these methods. After that we implement the Sinkhorn-Knopp algorithm on the simulated data in order to study the behaviour of the Frobenius score in terms of the step of the algorithm. From it, we will try to design a new phylogenetic inference method based on it.

4.1 Description of Simulated Data

We use simulated 4-taxon data generated by Fernández-Sánchez and Casanellas [8] under the general Markov model (GMM) and under the most general time-reversible model (GTR) introduced in Section 2.3.

4.1.1 Treespace

In order to generate the data, the 4-leaf tree of Figure 4.1 is assumed. The numbers $\{1, 2, 3, 4\}$ label the taxa, and a and b are the lengths of the edges according to the notation of Figure 4.1. The root r is located at the parent vertex of leaves 3 and 4.

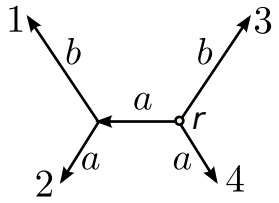


Figure 4.1: 4-leaf tree used in the generation of the simulated data.

The length a or b of the edges vary in the range $[0.01, 1.5]$ in steps of 0.02. Hence, we can consider a *treespace* (see [11]) with a and b as parameters (see figure 4.2). Then, for each pair (a, b), one hundred alignments are generated according to the corresponding tree evolving under GMM or GTR.

- **Simulations under GMM.** The data under GMM was generated using `GenNon-h` (see [12]). This software generates a random distribution of nucleotides at the root and, random

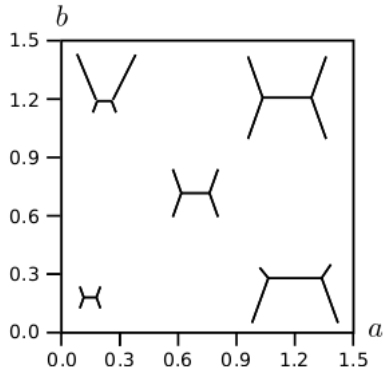


Figure 4.2: Treespace obtained from the tree represented in figure 4.1.

transition matrices with the expected amount of substitutions per site[1]. That is, for each value of a and b, the software generates 5 GMM transition matrices (one for each edge of the tree) taking into account the value of a and b.

- **Simulation under a time-homogeneous GTR model.** The data under GTR model was generated using `Seq-gen` (see [14]). The software generates a random distribution of nucleotides at the root, and random set of 5 transition matrices of the form $M_i = e^{Qt_i}$ for $i = 1, \ldots, 5$. Time-homogeneous GTR model means that the transition matrices are all generated by using the same matrix Q. What changes in every edge is the value of t_i which is related with the length of the corresponding edge (either a or b).

Following this scheme, 4 set of data were generated, according to the model (GMM or GTR) and to the length of the alignment (1000 or 10000).

The Felsenstein zone

There are two significant situations that must be highlighted because reconstruction methods use to perform poorly on them. On one hand, when there is a path connecting a pair of leaves with short length, then methods usually choose these leaves as neighbors instead of recovering the original topology. In the case of flattenings, when a is small (as explained in Section 2.5.4), the highest entries in the flattening along $13 \mid 24$ are distributed in the columns that represent that no mutation between 2 and 4 has occurred.
On the other hand, we can consider trees that are subject to *long-branch attraction*. That is, there are long edges on the tree and, because of this fact, the methods may not recover the correct tree topology. For instance, in Figure 4.3, the distance between the leaves 2 and 5 is very similar to the distance between 1 and 2. Hence, the number of different nucleotides is pretty similar (it is well known that the disagreement between two sequences tends to $\frac{3}{4}$ when the distance increases). That is, reconstruction methods may join leaf 2 with leaf 5 (for instance) instead of leaf 1, which is the topologically closer leaf.
In the case of the treespace considered, these two situations occur together in the so-called the *Felsenstein zone* (a is small, b is big). In the Felsenstein zone, reconstruction methods usually infer the topology induced by $13 \mid 24$ as the correct one, instead of choosing $12 \mid 34$.

[1]The length of any edge e is related with the determinant of the transition matrix associated to e, M_e.

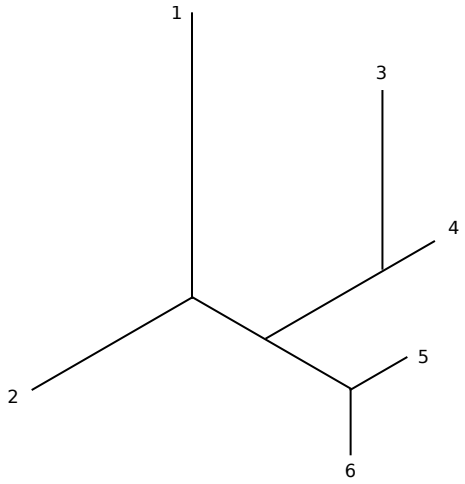


Figure 4.3: A tree affected with long branch attraction.

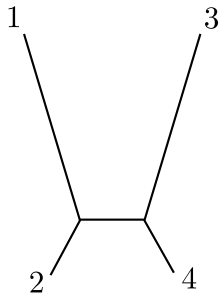


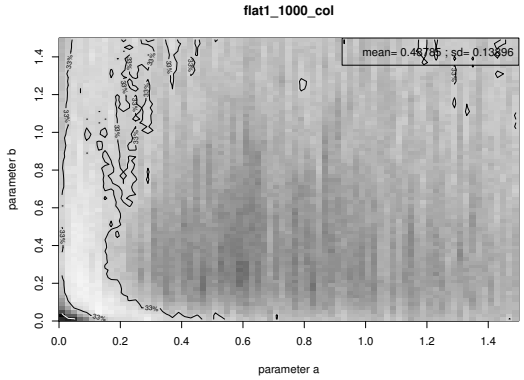
Figure 4.4: A tree of $\mathcal{T}_4$ affected with long branch attraction: the distance between the leaves 1 and 2 is larger that the distance between the leaves 2 and 4.

4.2 Results on treespace

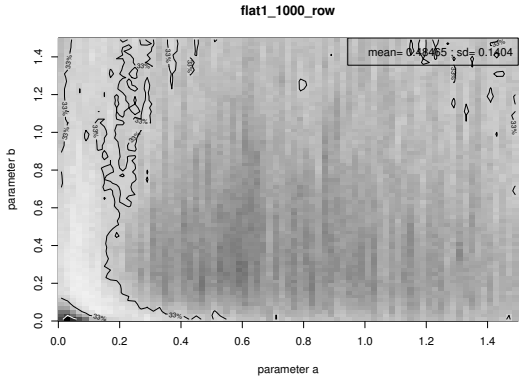
In this section, we analyze the results of the implementation of SK-method (Algorithm 3.2.1) on the simulated data just described.
The reader can see all the plots obtained in `https://copy.com/QsFfyVQ7UOVcYHlL` (see folders `SK-method_1000_GMM`, `SK-method_10000_GMM` and `SK-method_10000_GTR)`.
In Figure 4.5 we represent the success of this method on the treespace of figure 4.2. The darker a point is, the most success achieved. For instance, if a point is black, it means that the method has chosen the correct topology $(12 \mid 34)$ in all the alignments corresponding to those values of a and b.

Remark 4.2.1. By analyzing the performance of the method, we can note the following remarks:

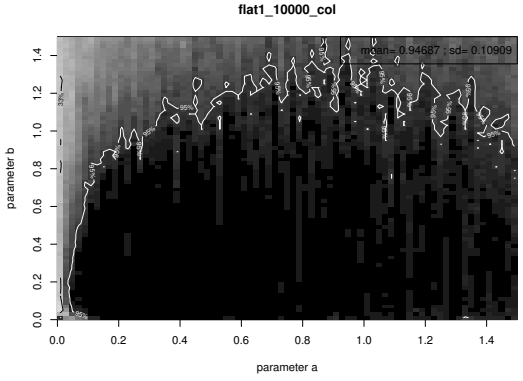
1. The method has significantly poor performance for length 1000 alignments. We observe that the method for 1000 base pairs alignments has very little success (average 48% of success). However, the results for 10000 base pairs alignments are much better (94,7% for GMM and 97,3% for GTR). We notice that the average success of the method starting either by columns or rows is the same for long alignments. For length 1000 alignments, the variation of the average success is not remarkable.



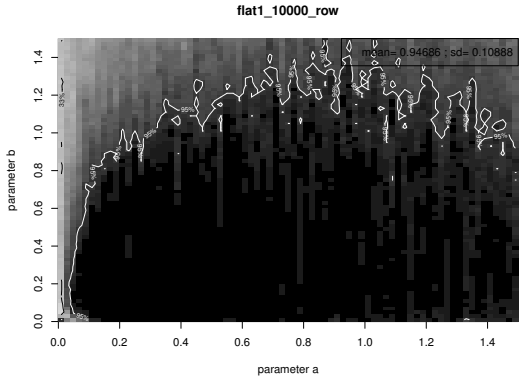
(a) GMM 1000, $\epsilon = 10^{-7}$, starting by columns.



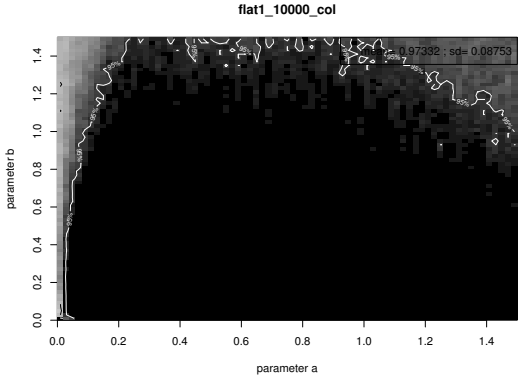
(b) GMM 1000, $\epsilon = 10^{-7}$, starting by rows.



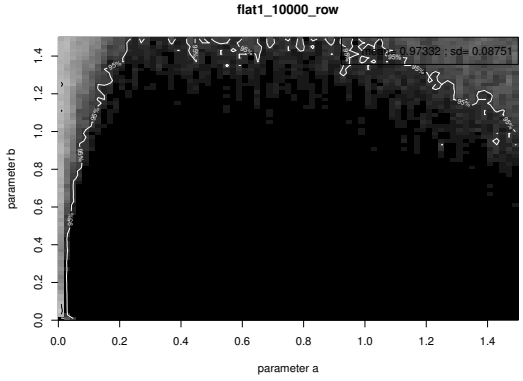
(c) GMM 10000, $\epsilon = 10^{-12}$, starting by columns.



(d) GMM 10000, $\epsilon = 10^{-12}$, starting by rows.



(e) GTR 10000, $\epsilon = 10^{-12}$, starting by columns.



(f) GTR 10000, $\epsilon = 10^{-12}$, starting by rows

Figure 4.5: Performance on the treespaces obtained when applying SK-method on simulated data allowing a maximum number of 100 iterations. On the left side we show the results obtained starting by columns and on the right side starting by rows. On the top, alignments of length 1000 generated under GMM are considered. On the middle, alignments of length 10000 generated under GMM. On the bottom, alignments of length 10000 generated under GTR.

2. The method has poor performance in the Felsenstein zone. In this case, when the method gives a wrong topology, usually chooses the 14 | 23.

3. The method also fails to recover the right topology for big values of b.

4. We can compare these results with other different reconstruction methods in the same treespace (from [8]). Although this method is based on `Erik+2` it shows a worst performance (see Table 4.1).

Method		1000 GMM	10000 GMM	10000 GTR
SK	Starting by columns	0.488 (0.14)	0.947 (0.11)	0.973 (0.09)
	Starting by rows	0.485 (0.14)	0.947 (0.11)	0.973 (0.09)
`ErikSVD`		**0.856** (0.21)	0.958 (0.13)	0.940 (0.22)
`Erik+2`		0.803 (0.17)	**0.971** (0.04)	**0.992** (0.04)
`NJ`		0.797 (0.18)	0.943 (0.09)	0.945 (0.10)
`ML`		0.736 (0.17)	0.754 (0.17)	0.980 (0.02)

Table 4.1: Average success of the SK-method, `ErikSVD`, `Erik+2`, neighbor-joining (`NJ`) and maximum likelihood (`ML`) on the data simulated as explained in Section 4.1. In parenthesis we show the standard deviation of the set of percentages of success of each method in each treespace.

Convergence of the method

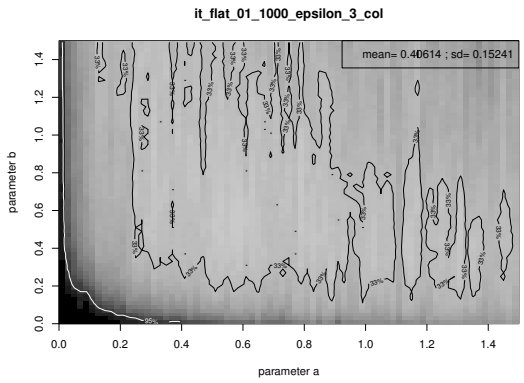
In order to understand the results obtained in the previous section, we first proceed to study the convergence of the Sinkhorn-Knopp algorithm applied to the flattenings. The flattenings could not fulfill the convergence condition of Theorem 3.1.10. To this aim, we plotted the convergence in the treespace of Figure 4.6. Given a value of $\epsilon > 0$ for each pair (a, b) the average number of iterations done by the algorithm (with a maximum number of iterations allowed equal to 100) is represented. That is, the darker a point is, the more iterations have been done in average.

Remark 4.2.2.
1. The convergence in the Felsenstein zone seems to be much slower than in the rest of the treespace. The number of iterations steps needed to reach some convergence criterium could be an indicator that a tree is in the Felsenstein zone.

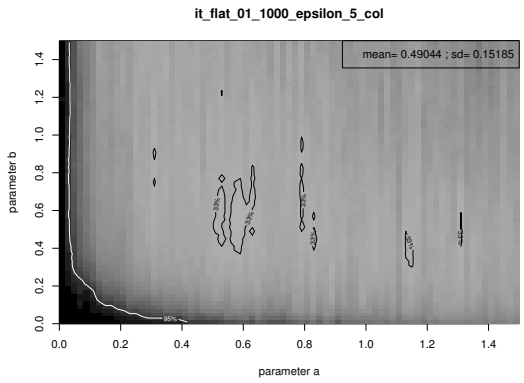
2. The sequences of length 1000 need more iterations in order to converge (an average of 41 for $\epsilon = 10^{-3}$ and 49 for $\epsilon = 10^{-5}$) while the sequences of length 10000 need, in average, 16 and 11 iterations for $\epsilon = 10^{-5}$ for GMM and GTR methods, respectively.

Variation of the Frobenius score along the algorithm

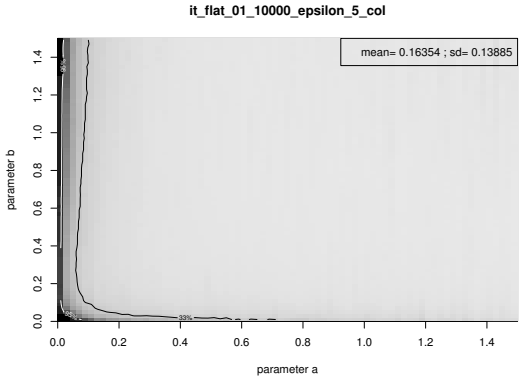
As we have seen in the previous section, the SK-method based on Sinkhorn-Knopp algorithm has a worst performance than `Erik+2`. In order to understand the reason and try to improve its performance, in this section we study how the Frobenius score evolves in each iteration of the algorithm. In order to do that, for each couple of values of (a, b), we computed the normalized average of the Frobenius scores of the flattenings at every step of the algorithm of



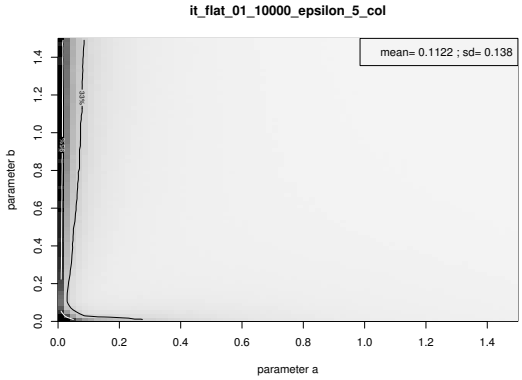
(a) GMM 1000 $\epsilon = 10^{-3}$, starting by columns



(b) GMM 1000 $\epsilon = 10^{-3}$, starting by columns



(c) GMM 10000 $\epsilon = 10^{-5}$, starting by columns



(d) GTR 10000 $\epsilon = 10^{-5}$, starting by columns

Figure 4.6: Iteration treespaces for $12 \mid 34$ when applying SK-method on simulated data allowing a maximum of 100 iterations in the process and different values of ϵ. All the figures are generated starting by columns. On the top, we show the results obtained for alignments of length 1000 generated under GMM for different values of ϵ. On the bottom, for alignments of length 10000 and $\epsilon = 10^{-5}$, on left side the alignments were generated under GMM and on the right side under GTR.

all the simulated data (100 different 4-taxon alignments). Recalling Remark 3.1.2, the ranks of the matrices do not change while applying the algorithm.
In this section we will show some figures that try to exemplify our observations. The reader can see all the plots in the Copy public folder.

Remark 4.2.3.

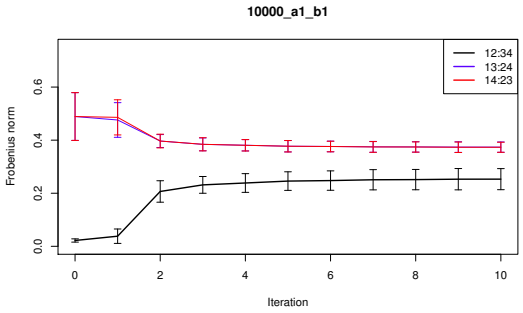
1. The behaviour and value of the Frobenius score while starting by columns or rows is very the same for $it > 2$. (see Figure 4.7). Furthermore, after the second iteration, the Frobenius score almost remains the same. This implies that the lines corresponding to the three topologies do not cross for $it > 2$.

2. In general, the plots for alignments of length 10000 are more regular that those corresponding to length 1000. Furthermore, in the Felsenstein zone seems that the algorithm does not reach the convergence criterium for the alignments of length 1000 (Figure 4.8 and

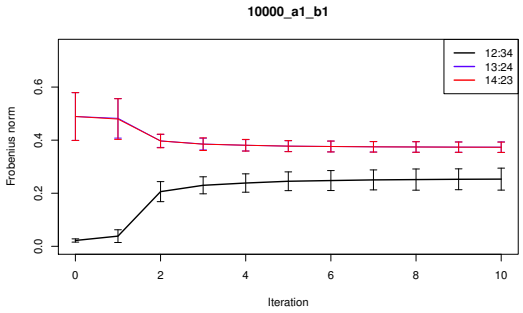
4.9). This fact could explain why we obtain differences between the average success when we start by columns or rows (see Remark 4.2.1). This fact is related with the convergence condition of the Sinkhorn-Knopp algorithm. The larger an alignment is, the less number of 0 entries had the flattening matrices and we expect the Sinkhorn-Knopp algorithm to converge. Moreover, as we previously note, in the Felsenstein zone the nonzero entries of a certain flattening matrices are distributed in the columns that represent that no mutation has been occurred. Hence, in the Felsensetein zone it is a hard to fulfill that flattening matrices have support.

3. In general for a big value of a, when the value of b grows we notice that the Frobenius score of the three topologies take similar values, i.e., the lines are closer (see Figure 4.8 and Figure 4.9). When the value of a is small, the lines get separated when b grows.

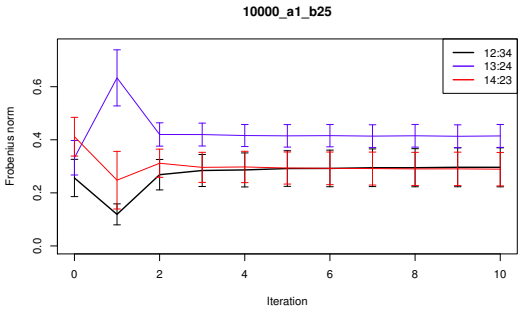
4. In the Felsenstein zone, we observe that for $it \geq 2$ the lines corresponding to $Flat_{12|34}(P)$ and $Flat_{14|23}(P)$ have similar values of the average Frobenius score while $Flat_{13|24}(P)$ has a greater value. Another interesting observation is the following: at $it = 0$ usually it is $Flat_{13|24}(P)$ that has a minimal Frobenius score. This implies that, from $it = 0$ to $it = 2$, the line corresponding to $Flat_{13|24}(P)$ has crossed the other two lines. We have observed that, if the iteration starts by columns, this crossing happens between $it = 0$ and $it = 1$, while if it starts by rows, the crossing is observed between $it = 1$ and $it = 2$. (see Figure 4.10). Notice that, for small values of a, when b grows the Frobenius score in $it = 0$ of $Flat_{13|24}(P)$ decreases while the Frobenius scores of $Flat_{12|34}(P)$ and $Flat_{14|23}(P)$ get closer.



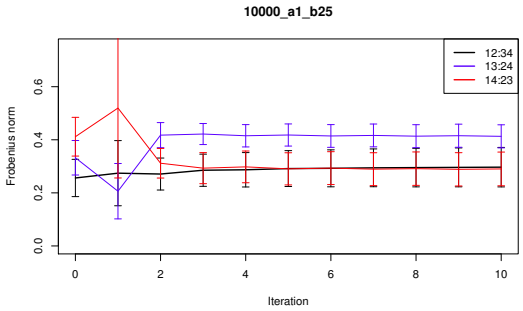
(a) $a = 1$, $b = 1$, starting by columns.



(b) $a = 1$, $b = 1$, starting by rows.

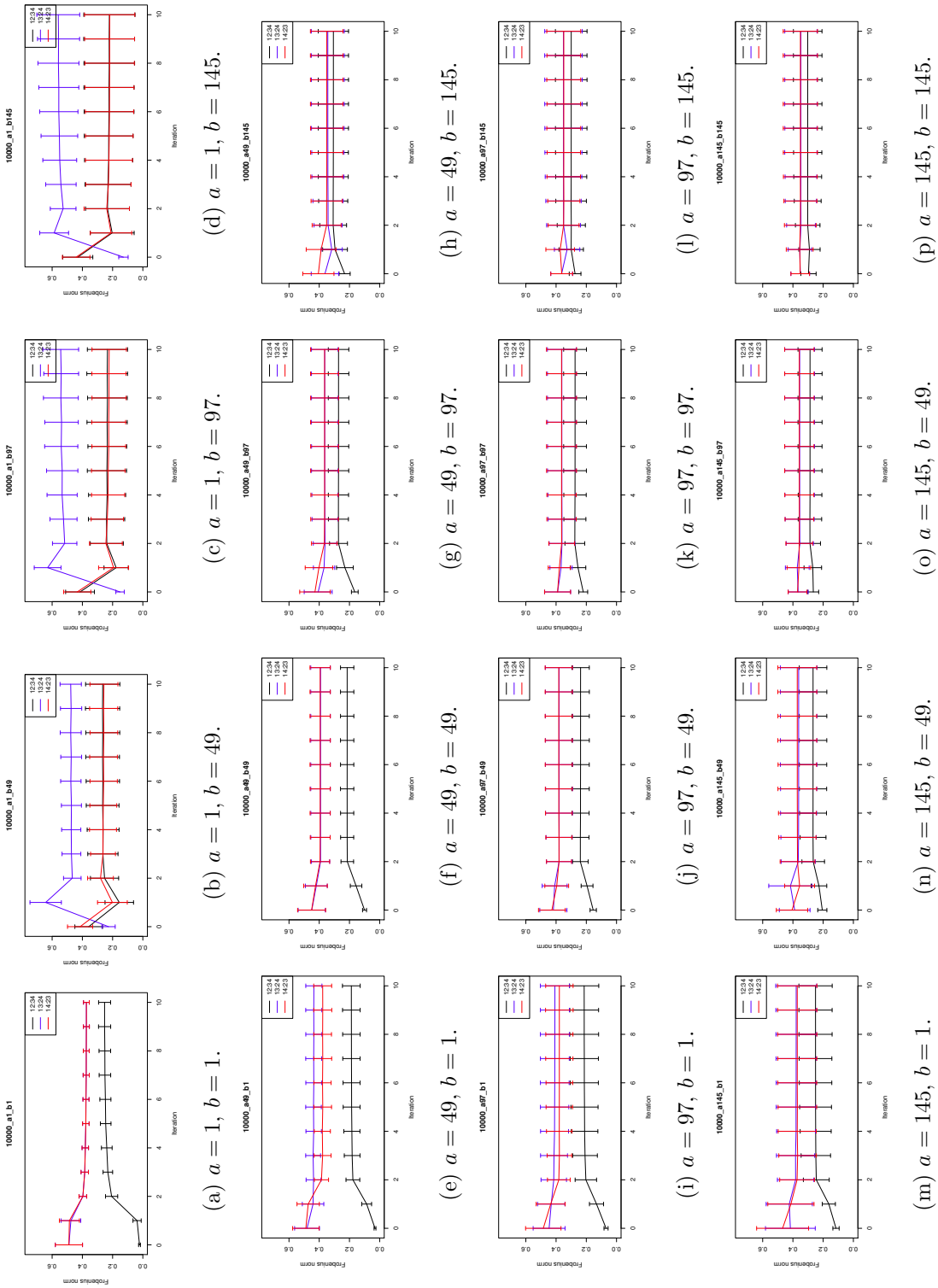


(c) $a = 1$, $b = 25$, starting by columns.



(d) $a = 1$, $b = 25$, starting by rows.

Figure 4.7: Normalized average Frobenius score of the SK-method for the three topologies of $\mathcal{T}_4$ ($\mathcal{T}_{12|34}$ in black, $\mathcal{T}_{13|24}$ in blue, $\mathcal{T}_{14|23}(P)$ in red). For some concrete values of a and b, 100 alignments of length 10000 are generated according to the tree of Figure 4.1 under GMM. The figures on the left side are obtained starting the method by columns, and on the right side by rows. The error bars of the figures denote the standard deviation of the data.



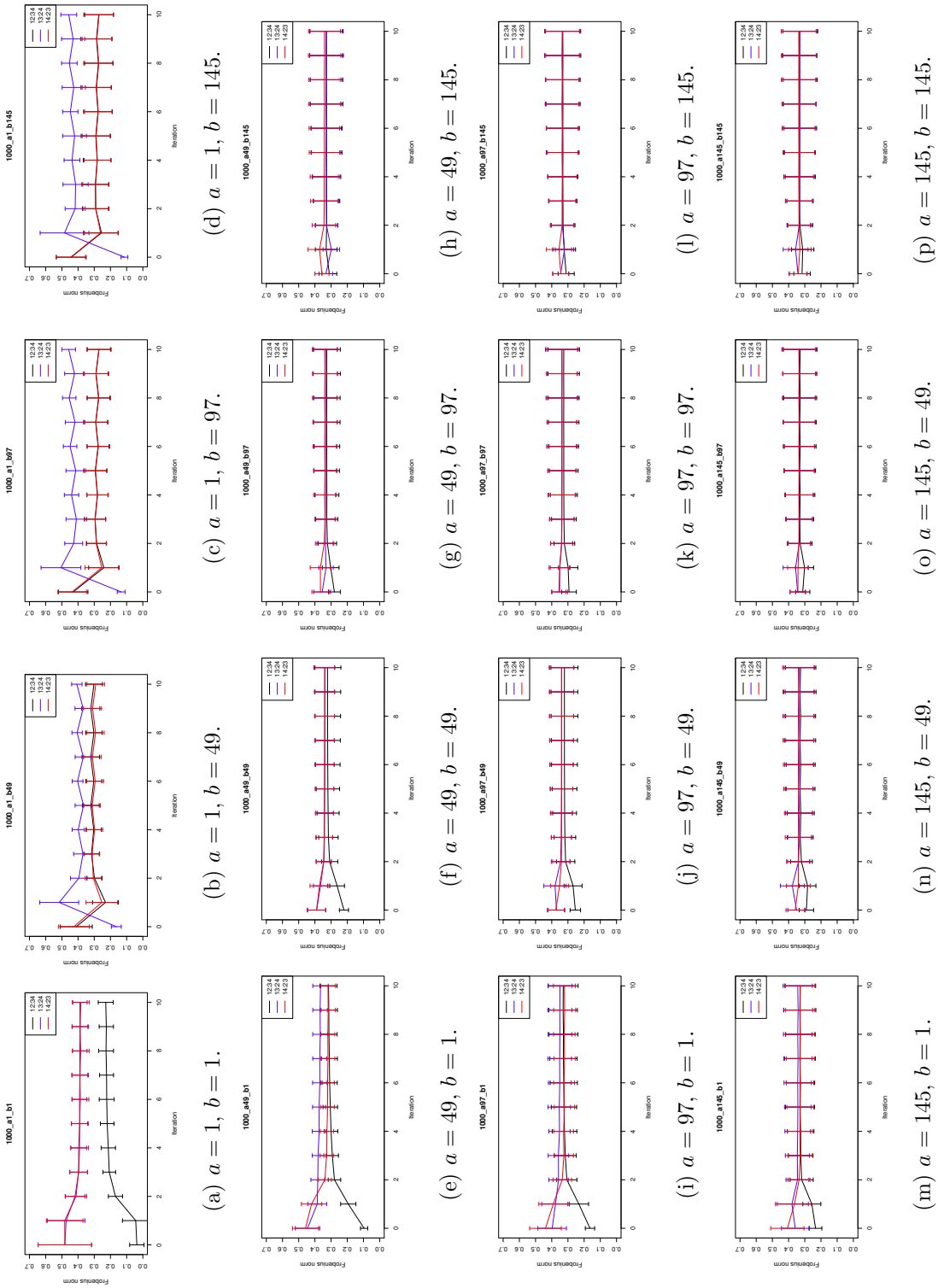
(a) $a = 1$, $b = 1$. (b) $a = 1$, $b = 49$. (c) $a = 1$, $b = 97$. (d) $a = 1$, $b = 145$.

(e) $a = 49$, $b = 1$. (f) $a = 49$, $b = 49$. (g) $a = 49$, $b = 97$. (h) $a = 49$, $b = 145$.

(i) $a = 97$, $b = 1$. (j) $a = 97$, $b = 49$. (k) $a = 97$, $b = 97$. (l) $a = 97$, $b = 145$.

(m) $a = 145$, $b = 1$. (n) $a = 145$, $b = 49$. (o) $a = 145$, $b = 49$. (p) $a = 145$, $b = 145$.

Figure 4.8: Normalized average Frobenius score of the SK-method for the three topologies of $\mathcal{T}_4$ ($\mathcal{T}_{12|34}$ in black, $\mathcal{T}_{13|24}$ in blue, $\mathcal{T}_{14|23}$ in red). For some concrete values of a and b, 100 alignments of length 10000 are generated according to the tree of Figure 4.1 under GMM. All the figures are obtained starting the method by columns. The error bars of the figures denote the standard deviation of the data.



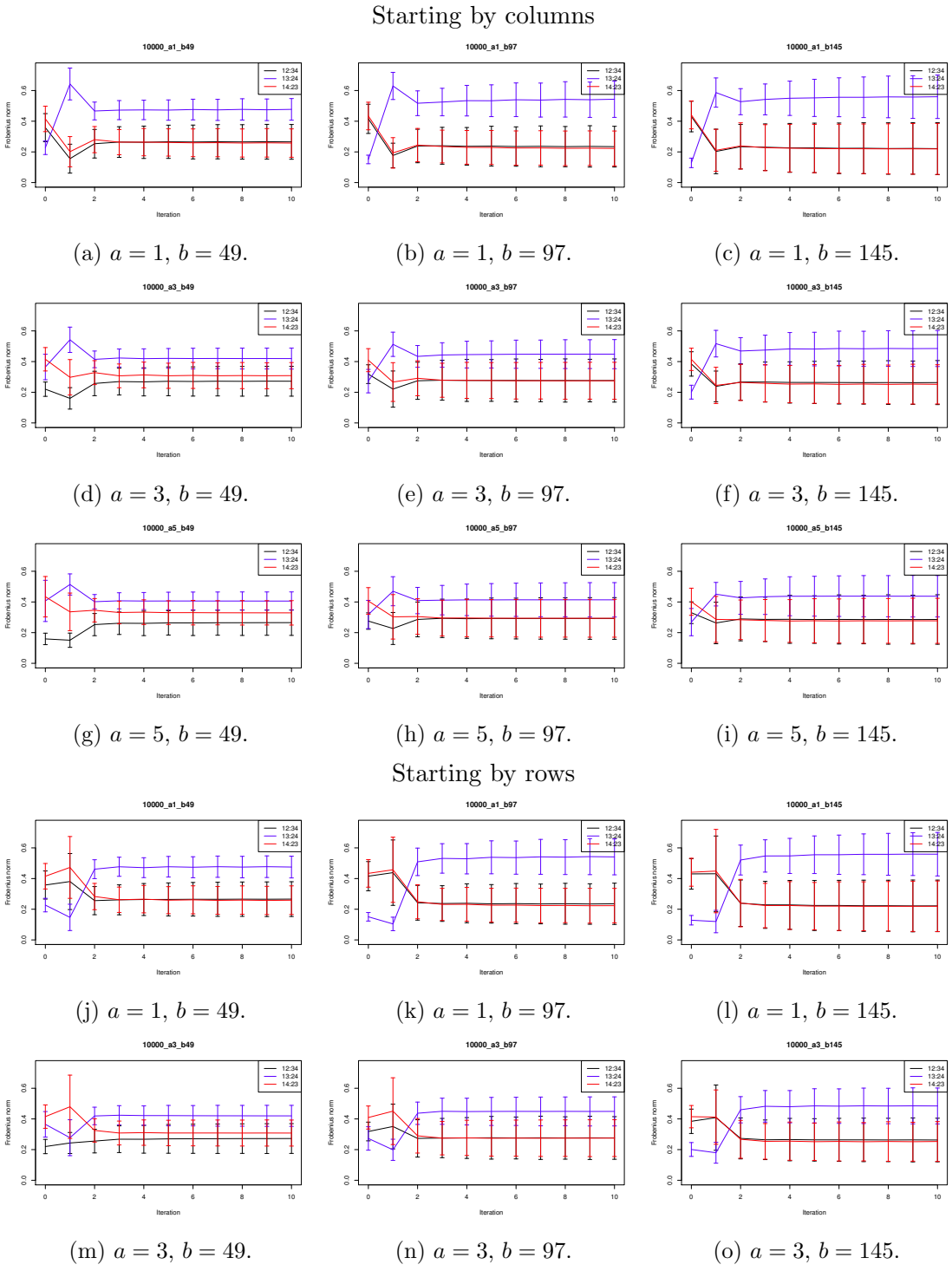
(a) $a = 1$, $b = 1$. (b) $a = 1$, $b = 49$. (c) $a = 1$, $b = 97$. (d) $a = 1$, $b = 145$.

(e) $a = 49$, $b = 1$. (f) $a = 49$, $b = 49$. (g) $a = 49$, $b = 97$. (h) $a = 49$, $b = 145$.

(i) $a = 97$, $b = 1$. (j) $a = 97$, $b = 49$. (k) $a = 97$, $b = 97$. (l) $a = 97$, $b = 145$.

(m) $a = 145$, $b = 1$. (n) $a = 145$, $b = 49$. (o) $a = 145$, $b = 49$. (p) $a = 145$, $b = 145$.

Figure 4.9: Normalized average Frobenius score in of the SK-method for the three topologies of $\mathcal{T}_4$ ($\mathcal{T}_{12|34}$ in black, $\mathcal{T}_{13|24}$ in blue, $\mathcal{T}_{14|23}$ in red). For some concrete values of a and b, 100 alignments of length 1000 are generated according to the tree of Figure 4.1 under GMM. All the figures are obtained starting the method by columns. The error bars of the figures denote the standard deviation of the data.



(a) $a = 1$, $b = 49$. (b) $a = 1$, $b = 97$. (c) $a = 1$, $b = 145$.

(d) $a = 3$, $b = 49$. (e) $a = 3$, $b = 97$. (f) $a = 3$, $b = 145$.

(g) $a = 5$, $b = 49$. (h) $a = 5$, $b = 97$. (i) $a = 5$, $b = 145$.

(j) $a = 1$, $b = 49$. (k) $a = 1$, $b = 97$. (l) $a = 1$, $b = 145$.

(m) $a = 3$, $b = 49$. (n) $a = 3$, $b = 97$. (o) $a = 3$, $b = 145$.

Figure 4.10: Normalized average Frobenius score of the SK-method for the three topologies of $\mathcal{T}_4$ ($\mathcal{T}_{12|34}$ in black, $\mathcal{T}_{13|24}$ in blue, $\mathcal{T}_{14|23}$ in red). For some concrete values of a and b in the Felsenstein zone, 100 alignments of length 10000 are generated according to the tree of Figure **??** under GMM. The block on the top is obtained starting by columns and on the bottom by rows. The error bars of the figures denote the standard deviation of the data.

Chapter 5

New methods for phylogenetic inference

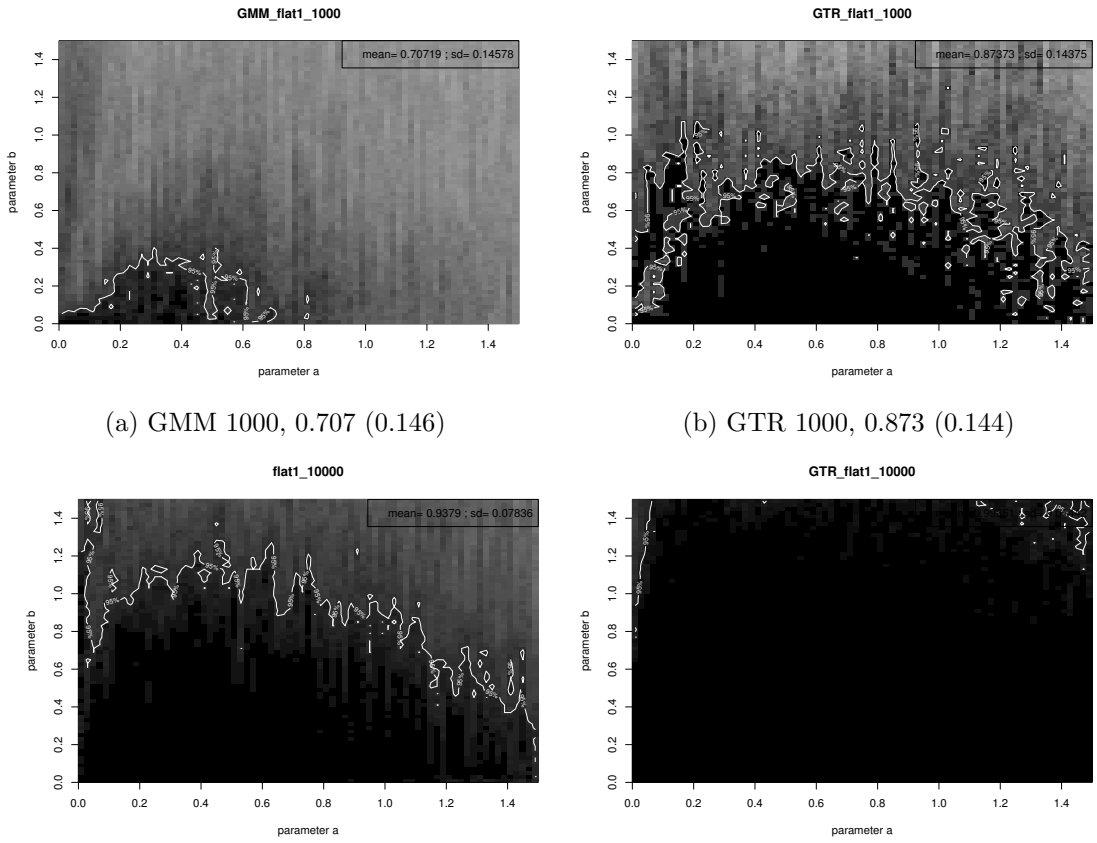
In this chapter, we design and test the performance of some new methods of phylogenetic inference based on the remarks of the previous chapter. The reader can see all the plots obtained in the Copy public folder in `https://copy.com/QsFfyVQ7UOVcYHlL` (see folder `New_method)`.

5.1 Crossing-criterium algorithm

First, in Remark 4.2.3 we noticed that for $it > 2$ the Frobenius scores almost do not vary. Hence, we just need to cut out the performance of the Sinkhorn-Knopp algorithm after the second iteration. Second, we observed that in the Felsenstein zone *crossings* between topologies occur. That is, the topology $13 \mid 24$ minimizes the Frobenius score in $it = 0$ and after one or two iterations (depending on whether we started by columns or rows) it has the maximum score. Based on this remark, we can modify SK-method (Algorithm 3.2.1) in order to have into account this fact in the Felsenstein zone. That is, we will ask the algorithm that in case of detecting a topology crossing the other two (either in the step 1 or 2) then that topology can not be chosen as the correct one.
Furthermore, since `ErikSVD` has a very good performance out of the Felsenstein zone (see [8]) we will establish the topology chosen by the algorithm as the one that minimizes the Frobenius score in $it = 0$.

Algorithm 5.1.1 (Crossing method)**.** Given a 4-taxon alignment,

- **Step 1**: Compute $d_4(Flat_{12|34}(P))$, $d_4(Flat_{13|24}(P))$ and $d_4(Flat_{14|23}(P))$.
- **Step 2**: For each matrix $Flat_{A|B}(P)$, apply two iterations of the Sinkhorn-Knopp algorithm (Algorithm 3.1.1) starting by rows and columns, i.e., compute $(Flat_{A|B}(P))_{cr}$ and $(Flat_{A|B}(P))_{rc}$.
- **Step 3**: If a crossing has been occurred for some flatenning $Flat_{A|B}(P)$, the corresponding topology $\mathcal{T}_{A|B}$ will not be considered in the next step.
- **Step 4**: Choose the bipartition with the minimum Frobenius score in $it = 0$.



(a) GMM 1000, 0.707 (0.146)

(b) GTR 1000, 0.873 (0.144)

(c) GMM 10000, 0.938 (0.078)

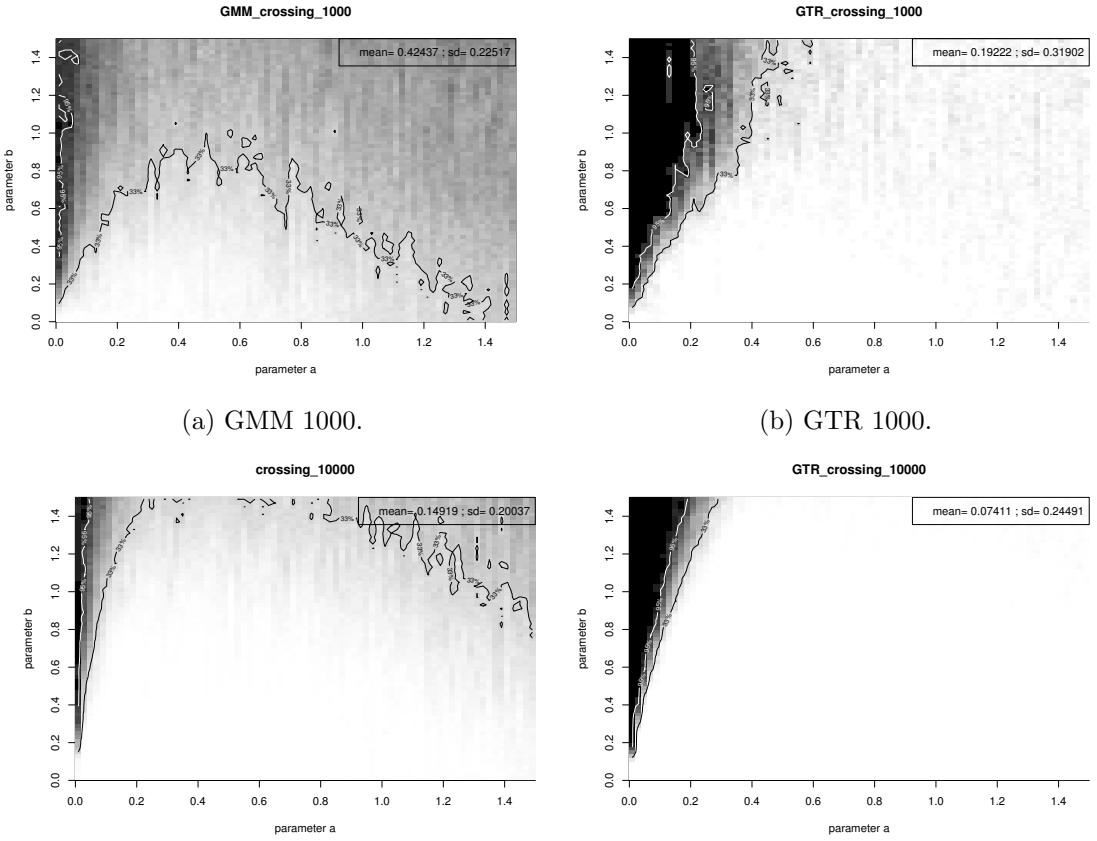
(d) GTR 10000, 0.997 (0.014)

Figure 5.1: Performance on the treespaces obtained when applying Crossing algorithm (Algorithm 5.1.1) on simulated data. On the left side we show the results obtained for alignments generated under GMM, and on the right side under GTR. On the top, alignments of length 1000 are considered, and on the bottom alignments of length 10000. The numbers in the captions represent the average success of Algorithm 5.1.1 (Crossing method). In parenthesis we show the standard deviation of the set of percentages of success.

Remark 5.1.2. In Figure 5.1 we represent the performance of the Algorithm 5.1.1 applied on simulated data according to the treespace of Figure 4.2. In Figure 5.2 we represent the number of times crossings between topologies have occurred.

1. In comparison with SK-method, the performance for length 1000 alignments (see Table 4.1) has improved significantly (0.48 for SK-method and 0.71 for Algorithm 5.1.1). For length 10000 alignments, the performances have not change regularly: we have a sightly better performance for GTR (0.997 vs 0.973) and sightly worse for GMM (0.938 vs. 0.973).

2. The performance in the Felsenstein zone is much better than the obtained for SK-method. However, it seems that it decline for big values of b and a. It might be that the crossing condition is too strong. That is, for data generated under GMM and for big values of b



(a) GMM 1000.

(b) GTR 1000.

(c) GMM 10000.

(d) GTR 10000.

Figure 5.2: Crossing condition treespaces obtained when applying Crossing algorithm (Algorithm 5.1.1) on simulated data. On the left side we show the results obtained for alignments generated under GMM and on the right side under GTR. On the top, alignments of length 1000 are considered and on the bottom alignments of length 10000.

and a, according to remark 4.2.3, the three flattenings have very close Frobenius score and could fulfill the crossing condition (see Figure 5.2, left side figures).

3. We notice that for data generated under GTR, the crossings occur mostly in Felsenstein zone. Moreover, for longer alignments under GTR, crossing condition demarcates precisely the Felsenstein zone (see Figure 5.2, right side figures). For the rest of simulated data, there are alignments that fulfill the crossing condition outside the Felsenstein zone gathered in the region with big values of b. This is the main reason why the method has a worse perform on this area than SK-method.

5.2 Crossings with a threshold difference greater than δ

In this section we design and test a new method which tries to correct the fact that crossings may occur for big values of b out of the Felsenstein zone because (according to Remark 5.1.2), in this region of the treespace, the Frobenius score of the three flattenings is very similar. Hence, a natural idea is asking that the jump done in the crossing must be bigger that a fixed value δ. In this case, we will say that a *δ-crossing* has occurred.

Algorithm 5.2.1 (δ-crossing method)**.** Given a 4-taxon data alignment and fixed some $\delta \in [0, 1]$,

- **Step 1**: Compute $d_4(Flat_{12|34}(P))$, $d_4(Flat_{13|24}(P))$ and $d_4(Flat_{14|23}(P))$.
- **Step 2**: For each matrix $Flat_{A|B}(P)$, apply two iterations of the Sinkhorn-Knopp algorithm (Algorithm 3.1.1) starting by rows and columns, i.e., compute $(Flat_{A|B}(P))_{cr}$ and $(Flat_{A|B}(P))_{rc}$.
- **Step 3**: If a crossing occurs for some flatenning $Flat_{A|B}(P)$ and the difference between the Frobenius score before and after the crossing is greater than δ, then the topology $\mathcal{T}_{A|B}$ will not be considered in the next step.
- **Step 4**: Choose the bipartition with the minimum Frobenius score in $it = 0$.

Remark 5.2.2. Since Frobenius scores are normalized, they only can take values between 0 and 1. If we choose $\delta = 0$ Algorithm 5.2.1 is the same than Algorithm 5.1.1.

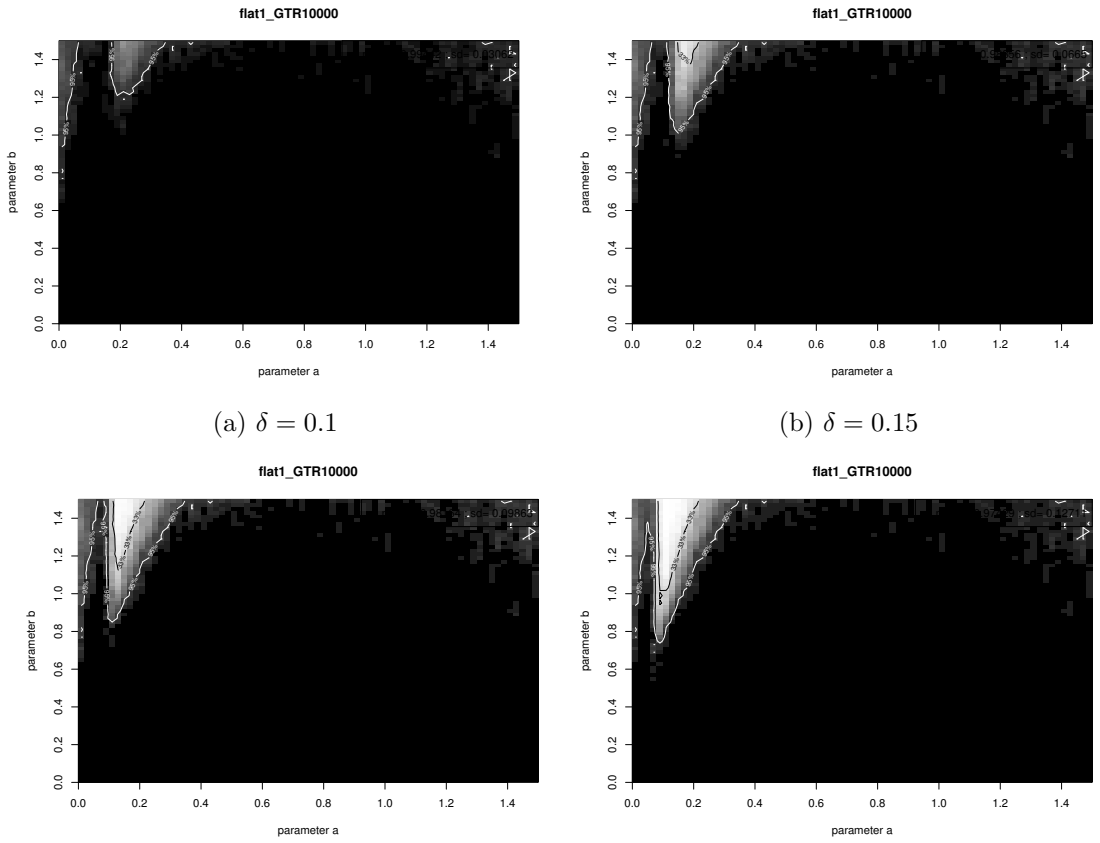
In order to avoid the cases where the Frobenius scores of the three topologies have similar values, we have implemented the algorithm for δ in the range $[0.1, 0.3]$ in steps of 0.5. The results of this implementation are summarized in Table 5.1.

Method	1000 GMM	10000 GMM	1000 GTR	10000 GTR
δ-crossing, $\delta = 0.10$	0.833 (0.116)	0.960 (0.048)	**0.862 (0.208)**	**0.994** (0.031)
δ-crossing, $\delta = 0.15$	0.867 (0.136)	0.975 (0.036)	0.846 (0.235)	0.984 (0.067)
δ-crossing, $\delta = 0.20$	**0.873** (0.156)	0.982 (0.038)	0.833 (0.254)	0.982 (0.099)
δ-crossing, $\delta = 0.25$	0.870 (0.172)	**0.983** (0.050)	0.825 (0.264)	0.974 (0.127)
δ-crossing, $\delta = 0.30$	0.866 (0.183)	0.981 (0.063)	0.817 (0.277)	0.967 (0.149)

Table 5.1: Average success of the δ-algorithm on the simulated data for different values of δ. In parenthesis we show the standard deviation of the set of percentages of success in each treespace.

Remark 5.2.3.

1. In comparison with the Crossing algorithm (Algorithm 5.1.1) we notice a better average performance for the simulated data under GMM (specially for alignments of 1000 nucleotides) and a sightly worst performance under GTR. Moreover, when the value of δ increases, the average success for the data under GMM first increases and then decreases, while for GTR it always goes down.
2. For data generated under GMM the method performs better when the value of δ is 0.20 for length 1000 alignments and for 0.25 for length 10000. However in this last case, the difference of average success for $\delta \in [0.20, 0.3]$ is very small. For data generated under



(a) $\delta = 0.1$ (b) $\delta = 0.15$

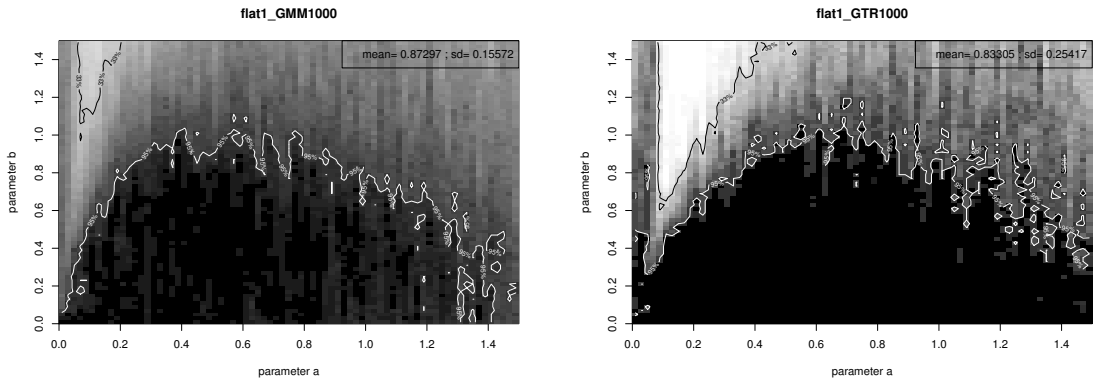
(c) $\delta = 0.2$ (d) $\delta = 0.25$

Figure 5.3: Performance on the treespaces obtained when applying δ-crossing method (Algorithm 5.2.1) on alignments of length 1000 simulated under GTR for different values of δ.

GTR, δ-crossing algorithm has a better performance for $\delta = 0.1$ for both length 1000 and length 1000 alignments.

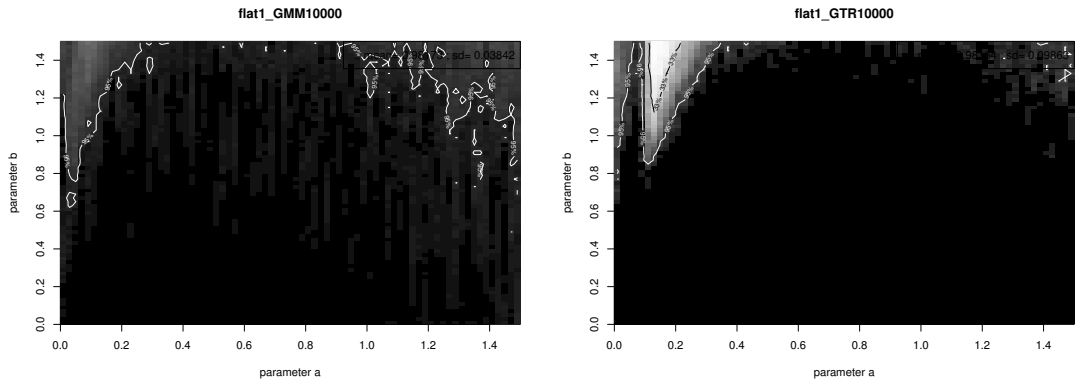
3. We notice that for all the simulated data, when the value of δ increases there is a region on the treespace (around $[0.15, 0.3] \times [0.85, 1.5]$) where the performances get worse (see Figure 5.3 for an example). This affects the value of the standard deviation of the average succes, which also increases its value when δ increases.

4. We notice that δ-crossings mostly occur in the Felsenstein zone (see Figure 5.5). For GTR, δ-crossings region demarcate precisely the Felsenstein zone (as expected for Remark 5.1.2). When the value of δ gets increased, we would expected that the δ-crossings region gets smaller because the condition is more restrictive. Indeed, we observe this behaviour in Figure 5.5. In comparison with Crossing algorithm, for data generated under GMM, δ-crossings region is concentrated on the Felsenstein zone while δ-crossings do not occur in other regions of the treespace when the value of δ is increased. However, δ-crossings region becomes really thin for big values of δ. Because all these remarks, we consider that δ-crossings may be a good indicator that a tree is in the Felsenstein zone.



(a) GMM 1000, 0.873 (0.156)

(b) GTR 1000, 0.873 (0.156)



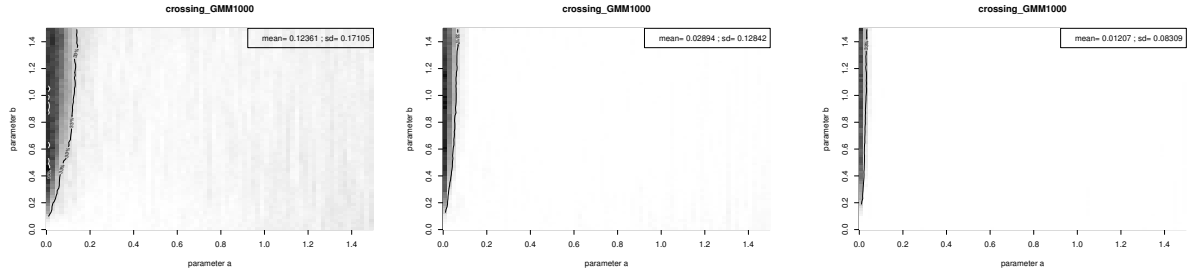
(c) GMM 10000, 0.982 (0.038)

(d) GTR 10000, 0.982 (0.099)

Figure 5.4: Performance on the treespaces obtained when applying δ-algorithm on simulated data allowing a maximum for $\delta = 0.2$. On the top, alignments of length 1000 generated under GMM are considered. On the middle, alignments of length 10000 generated under GMM. On the bottom, alignments of length 10000 generated under GTR.

5. Due to the previous remarks, we do consider that the best δ-crossing method overall is for $\delta = 0.2$. On one hand, average success for data generated under GMM have the maximum success around this value of δ and the average success for data generated under GTR is good and it has values not far away from the best (for $\delta = 0.1$). Furthermore, the δ-crossings region is fitting the Felsenstein zone and it is not as thin as for bigger values of δ. In Figure 5.4 we represent the performance of δ-crossing algorithm when $\delta = 0.2$.

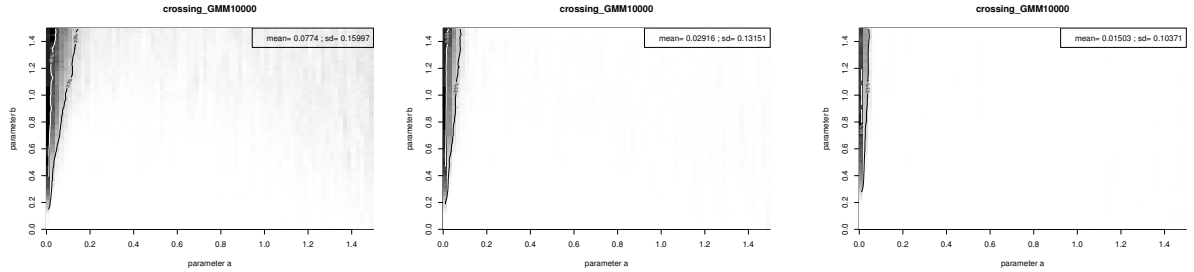
6. In Figure 5.4 we notice that the performances for big values of b have been improved in comparison with Crossing algorithm. That was the main motivation for the design of δ-crossing method.



(a) GMM 1000, $\delta = 0.1$.

(b) GMM 1000, $\delta = 0.2$.

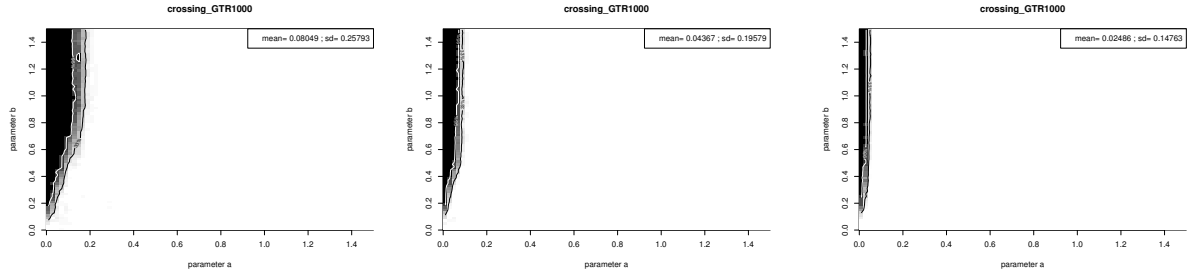
(c) GMM 1000, $\delta = 0.3$.



(d) GMM 10000, $\delta = 0.1$.

(e) GMM 10000, $\delta = 0.2$.

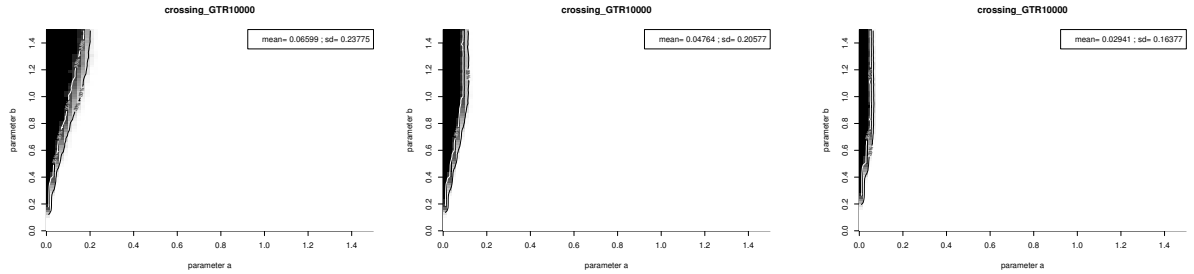
(f) GMM 10000, $\delta = 0.3$.



(g) GTR 1000, $\delta = 0.1$.

(h) GTR 1000, $\delta = 0.2$.

(i) GTR 1000, $\delta = 0.3$.



(j) GTR 10000, $\delta = 0.1$.

(k) GTR 10000, $\delta = 0.2$.

(l) GTR 10000, $\delta = 0.3$.

Figure 5.5: δ-crossing condition treespaces obtained when applying δ-crossing algorithm (Algorithm 5.2.1) on simulated data. On the left side we show the results obtained for alignments generated for $\delta = 0.1$, in the middle for $\delta = 0.2$ and on the right side for $\delta = 0.3$. Rows 1 and 2 correspond to data generated under GMM for short and long alignments respectively. Rows 3 and 4 correspond to data generated under GTR for short and long alignments respectively.

5.3 Crossing with relative distance

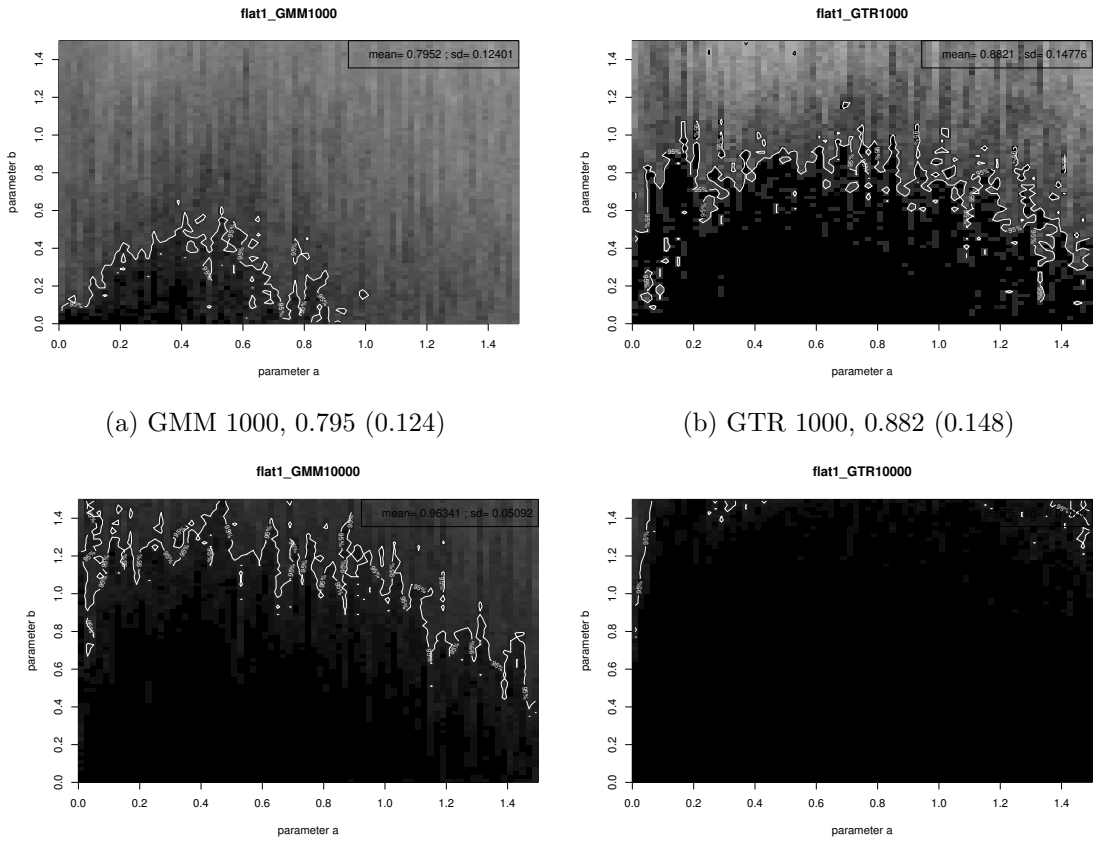
In the previous section we consider the δ-crossing algorithm: it looks for a crossing such that the difference of the Frobenius score of the flattening before and after the crossing is greater than a given threshold $\delta > 0$. The value of this parameter δ could be chosen in terms of our knowledge of the data.
However, we would like to design an inference method not involving external parameters. In Remark 4.2.3 we notice that in the Felsenstein zone the Frobenius score of $Flat_{12|34}(P)$ and $Flat_{14|23}(P)$ have similar values in $it = 1$ or $it = 2$ (starting by columns or rows respectively) while $Flat_{13|24}(P)$ has a greater value. Arising from this observation, we have designed the following algorithm.

Algorithm 5.3.1 (Dist-crossing method)**.** Given a 4-taxon alignment,

- **Step 1**: Compute $d_4(Flat_{12|34}(P))$, $d_4(Flat_{13|24}(P))$ and $d_4(Flat_{14|23}(P))$.
- **Step 2**: For each matrix $Flat_{A|B}(P)$, apply two iterations of the Sinkhorn-Knopp algorithm (Algorithm 3.1.1) to $(Flat_{A|B}(P))$ and rows.
- **Step 3**: If a crossing has been occurred for some flatenning $Flat_{A|B}(P)$, consider the flattening that has the middle value of the Frobenius score after the crossing. If this value is closer to the minimum value of the Frobenius score than to the maximum value, do not consider the topology $\mathcal{T}_{A|B}$ in the next step.
- **Step 4**: Choose the bipartition with the minimum Frobenius score in $it = 0$.

In Figure 5.6 we show the treespaces obtained applying Algorithm 5.3.1 on simulated data.

Remark 5.3.2.
1. This method performs better or equal than Crossing-algorithm for all simulated data. In comparison with δ-crossing method, it has a worse performance for data generated under GMM and sightly better under GTR.
2. Notice that for big values of b, it has a worse performance than the δ-crossing method.
3. The condition imposed in Step 3 of the algorithm occurs mostly in the Felsenstein zone for length 10000 alignments (see Figure 5.7). Notice the similarity between Figure 5.7 and Figure 5.2. The main difference is observed on the top-right part of the treespace for length 10000 alignments generated under GMM.



(a) GMM 1000, 0.795 (0.124)

(b) GTR 1000, 0.882 (0.148)

(c) GMM 10000, 0.963 (0.051)

(d) GTR 10000, 0.997 (0.014)

Figure 5.6: Performance on the treespaces obtained when applying Algorithm 5.3.1 on simulated data. On the top, alignments of length 1000 generated under GMM are considered. On the middle, alignments of length 10000 generated under GMM. On the bottom, alignments of length 10000 generated under GTR.

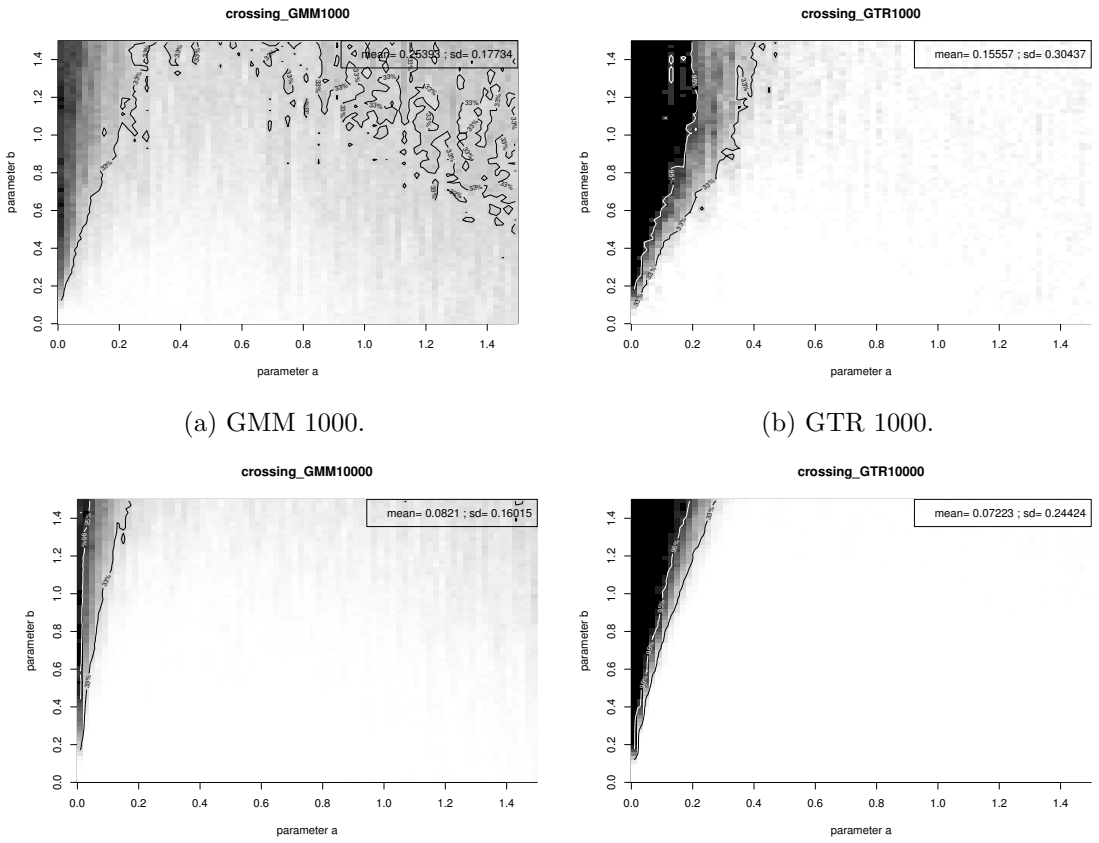
5.4 Results on treespace

In this section we compare the performance of the new methods for phylogenetic inference described in the previous sections on data generated according to the treespace of Figure 4.2.

Table 5.2 shows the average success of the methods for phylogenetic reconstruction considered in this work.

For the simulated data generated under GMM we notice that the method with the best performance is δ-crossing, with $\delta = 0.2$ for length 1000 alignments and $\delta = 0.25$ for length 10000 alignments. However, the difference of performance for length 10000 (0.982 vs. 0.983) is negligible and the standard deviation for $\delta = 0.2$ is smaller, so we choose the δ-crossing with $\delta = 0.2$ as our best candidate method for phylogenetic reconstruction.

For the data generated under GTR, we observe that the Crossing method and Dist-crossing method have the best performance for length 10000 alignments. Both methods have a better performance than `Erik+2`.



(a) GMM 1000.

(b) GTR 1000.

(c) GMM 10000.

(d) GTR 10000.

Figure 5.7: Crossing condition treespaces obtained when applying Algorithm 5.3.1 on simulated data. On the left side we show the results obtained for alignments generated under GMM and on the right side under GTR. On the top, alignments of length 1000 are considered and on the bottom alignments of length 10000

5.5 Results on arbitrary quartet trees

In the previous section we observed the good performance of the new methods presented in this project for the simulated data on the treespace (see Section 4.1.1). However, this is not a realistic situation since we have considered a 4-leaf tree (Figure 4.1) with very concrete lengths on its edges for the generation of the data.

In order to test the methods for phylogenetic reconstruction in a more general situation, we generated two sets of 10000 alignments each under GMM and GTR assuming that the tree topology is $\mathcal{T}_{12|34}$. The length of the edges take random values between 0 and 1 (for the first set) and between 0 and 3 for the second set.

In this last section, we test the performance of the Crossing method, δ-crossing method for $\delta = 0.2$, Dist-crossing, `ErikSVD` and `Erik+2` methods on simulated data generated under a quartet tree with random branch length. Also, for the second set of data we also tested `NJ` and `ML`. In Table 5.3 and 5.4 we show the number of success obtained for the methods for the two

Method	1000 GMM	10000 GMM	1000 GTR	10000 GTR
Crossing	0.707 (0.15)	0.938 (0.08)	0.874 (0.14)	**0.997** (0.01)
δ-crossing, $\delta = 0.10$	0.833 (0.12)	0.960 (0.05)	0.862 (0.21)	0.994 (0.03)
δ-crossing, $\delta = 0.15$	0.867 (0.14)	0.975 (0.04)	0.846 (0.24)	0.984 (0.07)
δ-crossing, $\delta = 0.20$	**0.873** (0.16)	0.982 (0.04)	0.833 (0.25)	0.982 (0.10)
δ-crossing, $\delta = 0.25$	0.870 (0.17)	**0.983** (0.05)	0.825 (0.26)	0.974 (0.13)
δ-crossing, $\delta = 0.30$	0.866 (0.18)	0.981 (0.06)	0.817 (0.28)	0.967 (0.15)
Dist-crossing	0.795 (0.12)	0.963 (0.05)	0.882 (0.15)	**0.997** (0.01)
`ErikSVD`	0.856 (0.21)	0.958 (0.13)	0.796 (0.30)	0.940 (0.22)
`Erik+2`	0.803 (0.17)	0.971 (0.04)	0.843 (0.19)	0.992 (0.04)
`NJ`	0.797 (0.18)	0.943 (0.09)	0.805 (0.20)	0.945 (0.10)
`ML`	0.736 (0.17)	0.754 (0.17)	**0.934** (0.06)	0.980 (0.02)

Table 5.2: Average success of the Crossing method, δ-crossing, Dist-crossing, `ErikSVD`, `Erik+2`, neighbor-joining (`NJ`) and maximum likelihood (`ML`) on the data simulated as explained in Section 4.1. In parenthesis we show the standard deviation of the set of percentages of success of each method in each treespace.

sets of 10000 alignments with random branches. As it could be expected, due to long branch attraction, the performance for the first set of generated data is better than for the second set of data. Also notice that length 10000 alignments has better performance than alignments with length 1000.

Remark 5.5.1.

1. The method presented in this project with a better performance in all cases is the δ-crossing method with $\delta = 0.2$.

2. Since the new methods of this memoir are based on the same ideas than `ErikSVD` and `Erik+2` we are specially interested to compare their performance. For data generated under GMM, the δ-crossing method has a better performance than `Erik+2` except in one case: alignments with length 10000 and random branches with length between 0 and 3. For data generated under GTR, δ-crossing method and `Erik+2` have similar number of success (8916 vs. 8931 and 9638 vs. 9693) while for the second set of data δ-crossing has a better performance (6579 vs. 6224 and 8105 vs. 8030).

3. For all data generated under GMM we observe that the δ-crossing method performs better than `ML`. However, in this case `NJ` is clearly the best method. For data generated under GTR, `ML` is the best method. In this case both `ML` and `NJ` performs better than δ-crossing.

Method	1000 GMM	10000 GMM	1000 GTR	10000 GTR
Crossing	7140	9385	7959	9527
δ-crossing, $\delta = 0.20$	9147	9688	8916	9638
Dist-crossing	7986	9542	8336	9580
`ErikSVD`	9399	9798	9124	9678
`Erik+2`	8729	9672	8931	9693
`NJ`	**9420**	**9840**	9260	9830
`ML`	8484	8691	**9670**	**9925**

Table 5.3: Number of success obtained for the Crossing method, δ-crossing, Dist-crossing, `ErikSVD`, `Erik+2`, neighbor-joining (`NJ`) and maximum likelihood (`ML`) on the 10000 alignments simulated with random branches with length between 0 and 1.

Method	1000 GMM	10000 GMM	1000 GTR	10000 GTR
Crossing	4675	6350	5797	7883
δ-crossing, $\delta = 0.20$	6378	7618	6579	8105
Dist-crossing	5413	6909	6041	7940
`ErikSVD`	6468	7762	6669	8157
`Erik+2`	6125	7973	6224	8030
`NJ`	**7520**	**8423**	7226	8275
`ML`	5014	5136	**7819**	**9234**

Table 5.4: Number of success obtained for the Crossing method, δ-crossing, Dist-crossing, `ErikSVD`, `Erik+2`, neighbor-joining (`NJ`) and maximum likelihood (`ML`) on the 10000 alignments simulated with random branches with length between 0 and 3.

Chapter 6

Conclusions

In this project, we have studied the Sinkhorn-Knopp algorithm ([17]) as a tool to normalize the flattening matrices arising from the sequences of alignment of four species. We focus on the case of four species because 4-leaf trees may be considered as building blocks to construct bigger trees by means of the so-called quartet methods ([15]). The application of the Sinkhorn-Knopp algorithm in the phylogenetic framework gives rise to a new phylogenetic reconstruction method, that we have called SK-method in the memoir. We have tested the performance of this method on the treespace introduced by Huelsenbeck (see [11]). From this, we have derived a number of conclusions.

- Due to the nature of the phylogenetic reconstruction problem, we cannot assume that the Sinkhorn-Knopp algorithm converges when applied to the flattening matrices associated to an alignment of nucleotide sequences. This is specially true when the length of the alignment is short (1000), as there will be a big number of null entries, against the assumption for convergence of Theorem 3.1.10.

- In spite of the lack of convergence in some cases, the simulations carried out seem to show a regular behaviour of the distances score after the first iterations of the method. This fact suggest that the convergence issue may not be very relevant for our purposes.

- In general, the SK-method performs poorly for length 1000. For length 10000, the results obtained improve considerably and are comparable to the results obtained by other methods based on similar ideas (`ErikSVD` or `Erik+2`). In this sense, it is important to note that true alignments of biological species use to be much longer than 1000 or 10000 sites.

- The methods based on the flattening matrices attains the best results in the simulations carried on Huelsenbeck's treespace beating popular reconstruction methods as Neighbour-Joining or Maximum-likelihood (see Table 4.1)

In order to improve the performance of this method, we have taken into account different aspects of the algorithm, as the speed of convergence of the sequence of matrices obtained. By studying the evolution of distances of the matrices associated to the three topologies, different patterns depending on the relative lengths of the branches of the trees can be noticed. One of these patterns, called *crossing* in the memoir, seems to detect the situation where a certain topology is wrongly chosen due to the small number of mutations between some of the leaves, i.e., the

so-called *Felsensetein zone*. With the aim to improve the performance in the Felsenstein zone, we considered the crossings where the difference between the Frobenius score after and before the crossing was greater thant a given value δ. In the Figure 5.5 we observe that δ-crossings mostly occur in the Felsenstein zone so it could be considered as an indicator to identify when a tree in the treespace belongs to this region. All this information reveals to be useful to prevent wrong inference in the simulation studies carried out in Chapter 5. From these studies, we derive the following conclusions:

- The methods *Crossing* and *δ-crossing* obtain good results when appied to arbitrary quartet trees. The first seems specially indicated to deal with data generated under the GTR model, while the second seems to perform better on data generated under the GMM (see Table 5.3).
- On data generated under trees with random length branches, the best-performing method among the new methods studies in this project is δ-crossings taking $\delta = 0.2$, which gives really good results. In this case, for data generated under GMM the performance is better than `Erik+2` in all cases except for one (see Tables 5.3 and 5.4). For data generated under GTR, we notice that `Erik+2` performs sightly better for the first set of generated data (see Table 5.3) while δ-crossing with $\delta = 0.2$ does it for the second set of generated data (Table 5.4), with a greater difference of number of success than in the previous case.
- The performance of these methods is not as good as that of `ML` for GTR, or `NJ` for GMM (see Tables 5.3 and 5.4).

Because of all these remarks, we think the approach based on the Sinkhorn-Knopp algorithm can be useful to improve the performance of `ErikSVD` or `Erik+2`, as it provides information not considered by these two methods. All these methods based on flattening matrices are specially indicated to deal with data that fit the situations represented in Huelsenbeck treespace (long-branch attraction). As possible lines of future research, we would like to point out:

- Study the performance on real data where the "true" phylogenetic tree is considered to be known.
- Test the method on data generated under more realistic models, as mixture data. Compare the method with other popular methods, as maximum parsimony.

Finally, all the new methods tested here have been programmed using `C++` and are available in `https://copy.com/QsFfyVQ7UOVcYHlL`.

Bibliography

[1] ES Allman, JA Rhodes, *Mathematical models in biology, an introduction*, Cambridge University Press, January 2004, ISBN 0-521-52586-1).

[2] ES Allman, JA Rhodes, *Phylogenetic invariants*. In O. Gascuel, and MA Steel, editors, *Reconstructing Evolution*. Oxford University Press.

[3] P. Buneman, *The recovery of trees from measures of dissimilarity*, in: E.U. Press (Ed.), Mathematics in the Archaeological and Historical Sciences, pp. 387-395.

[4] M. Casanellas, J. Fernández-Sánchez,, *Relevant phylogenetic invariants of evolutionary models*, J. Math. Pure. Appl. 96:207-229 (2010).

[5] JW. Demmel *Applied Numerical Linear Algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[6] R. Durbin, S. Eddy, A. Krogh, G. Mitchison, *Biologial sequence analysis. Probabilistic models of proteins and nucleic acids*, Cambridge University Press, 1998.

[7] N. Eriksson, *Tree construction using singular value decomposition*, in Algebraic statistics for computational biology, pages 347?358. Cambridge Univ. Press, New York, 2005.

[8] J. Fernández-Sánchez, M. Casanellas, *Invariant Versus Classical Quartet Inference When Evolution is Heterogeneous Across Sites and Lineages*, to appear in Systematic Biology (2016). DOI:10.1093/sysbio/syv086.

[9] O. Gascuel, editor *Mathematics of Evolution and Phylogeny*, Oxford University Press, 2005.

[10] Gu X, Li WH., *Bias-corrected paralinear and LogDet distances and tests of molecular clocks and phylogenies under nonstationary nucleotide frequencies*, Mol Biol Evol. 1996 Dec;13(10):1375-83.

[11] J. P. Huelsenbeck, *Performance of phylogenetic methods in simulation*, Syst. Biol., 44: 17?48, 1995.

[12] A. M. Kedzierska, M. Casanellas, *Gennon-h: Generating multiple sequence alignments on nonhomogeneous phylogenetic trees*, BMC Bioinformatics, 13(1):216, 2012.

[13] P. A. Knight, *The Sinkhorn-Knopp algorithm: convergence and applicactions*, SIAM Journal on Matrix Analysis and Applications, 2008.

[14] A. Rambaut, N. Grassly, *Seq-Gen: An application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees*, Comput. Appl. Biosci., 1997. 13:235?238.

[15] V. Ranwez, O. Gascuel, *Quarted-based phylogenetic inference: Improvements and limits.*, Molecular Biology and Evolution, 18(6):1103-1116, 2001.

[16] N. Saitou, M. Nei *The neighbor joining method: a new method for reconstructing phylogenetic trees*, Molecular Biology and Evolution, 4(4):406?425, 1987.

[17] R. Sinkhorn, *A relationship between arbitrary positive matrices and doubly stochastic matrices*, Ann. Math. Statist. 35 (1964), 876-879.

[18] R. Sinkhorn, P. Knopp, *Concerning nonnegative matrices and doubly stochastic matrices*, Pacific Journal of Mathematics Vol. 21, No. 2 (1967).

[19] K. Strimmer, A. von Haeseler, *Quartet puzzling: A quartet maximum likelihood method for reconstructing tree topologies.* Mol. Biol. Evol., 13:964-960, 1996.

[20] J. A. Studier, K. J. Keppler, *A note on the neighbor-joining algorithm of Saitou and Nei*, Molecular Biology and Evolution,, 5(6):729-731, November 1988.

[21] S. Tavaré *Some probabilistic and statistical problems in the analysis of DNA sequences*, Lectures on Mathematics in the Life Sciences (American Mathematical Society) 17, 57-86, 1986.

[22] S.J. Willson, *Building phylogenetic trees from quartets by using local inconsistency measures.* Molecular Biology and Evolution, 16(5):685, 1999.